Очередь — это линейный список информации, работа с которой происходит по принципу "первым пришел — первым вышел" (first-in, first-out); этот принцип (и очередь как структура данных) иногда еще называется FIFO[[1]](http://www.chitay.org/c/22/2201.htm" \l "11). Это значит, что первый помещенный в очередь элемент будет получен из нее первым, второй помещенный элемент будет извлечен вторым и т.д. Это единственный способ работы с очередью; произвольный доступ к отдельным элементам не разрешается.

Очереди очень часто встречаются в реальной жизни, например, около банков или ресторанов быстрого обслуживания. Чтобы представить себе работу очереди, давайте введем две функции: qstore() и qretrieve() (от "store"— "сохранять", "retrieve" — "получать"). Функция qstore() помещает элемент в конец очереди, а функция qretrieve() удаляет элемент из начала очереди и возвращает его значение. В табл. 22.1 показано действие последовательности таких операций.

|  |  |
| --- | --- |
| **Таблица 22.1. Работа очереди** | |
| ***Действие*** | ***Содержимое очереди*** |
| qstore(A) | A |
| qstore(B) | А В |
| qstore(C) | A B C |
| qretrieve() возвращает А | В С |
| qstore(D) | B C D |
| qretrieve() возвращает В | C D |
| qretrieve() возвращает С | D |

Следует иметь в виду, что операция извлечения удаляет элемент из очереди и уничтожает его, если он не хранится где-нибудь в другом месте. Поэтому после извлечения всех элементов очередь будет пуста.

В программировании очереди применяются при решении многих задач. Один из наиболее популярных видов таких задач — симуляция. Очереди также применяются в планировщиках задач операционных систем и при буферизации ввода/вывода.

Чтобы проиллюстрировать работу очереди, мы напишем простую программу планирования встреч. Эта программа позволяет сохранять информацию о некотором количестве встреч; потом по мере прохождения каждой встречи она удаляется из списка. Для упрощения описание встреч ограничено 255 символами, а количество встреч — произвольным числом 100.

При разработке этой простой программы планирования необходимо прежде всего реализовать описанные здесь функции qstore() и qretrieve(). Они будут хранить указатели на строки, содержащие описания встреч.

#define MAX 100

char \*p[MAX];

int spos = 0;

int rpos = 0;

/\* Сохранение встречи. \*/

void qstore(char \*q)

{

if(spos==MAX) {

printf("Список переполнен\n");

return;

}

p[spos] = q;

spos++;

}

/\* Получение встречи. \*/

char \*qretrieve()

{

if(rpos==spos) {

printf("Встреч больше нет.\n");

return '\0';

}

rpos++;

return p[rpos-1];

}

Обратите внимание, что этим двум функциям требуются две глобальные переменные: spos, в которой хранится индекс следующего свободного места в списке, и rpos, в которой хранится индекс следующего элемента, подлежащего выборке. С помощью этих функций можно организовать очередь данных другого типа, просто поменяв базовый тип обрабатываемого ими массива.

Функция qstore() помещает описания новых встреч в конец списка и проверяет, не переполнен ли список. Функция qretrieve() извлекает встречи из очереди, если таковые имеются. При назначении встреч увеличивается значение переменной spos, а по мере их прохождения увеличивается значение переменной rpos. По существу, rpos "догоняет" spos в очереди. На рис 22.1 показано, что может происходить в памяти при выполнении программы. Если rpos и spos равны, назначенные события отсутствуют. Даже несмотря на то, что функция qretrieve() не уничтожает хранящуюся в очереди информацию физически, эту информацию можно считать уничтоженной, так как повторно получить доступ к ней невозможно.

|  |
| --- |
| **Начальное сосотояние очереди**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | | | | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos  **qstore('A')**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | A | | | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos  **qstore('B')**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | A | B | | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos  **qretrive()**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | | B | | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos  **qretrive()**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | | | | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos  **qstore('A')**  ↓spos  +---+---+---+---+---+---+---+---+---+---+---+  | | | C | | | | | | | | |  +---+---+---+---+---+---+---+---+---+---+---+  ↑rpos |
| *Рис. 22.1. Индекс выборки "догоняет" индекс вставки* |

Текст программы простого планировщика встреч целиком приведен ниже. Вы можете доработать эту программу по своему усмотрению.

/\* Мини-планировщик событий \*/

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

#define MAX 100

char \*p[MAX], \*qretrieve(void);

int spos = 0;

int rpos = 0;

void enter(void), qstore(char \*q), review(void), delete\_ap(void);

int main(void)

{

char s[80];

register int t;

for(t=0; t < MAX; ++t) p[t] = NULL; /\* иницилизировать массив

пустыми указателями \*/

for(;;) {

printf("Ввести (E), Список (L), Удалить (R), Выход (Q): ");

gets(s);

\*s = toupper(\*s);

switch(\*s) {

case 'E':

enter();

break;

case 'L':

review();

break;

case 'R':

delete\_ap();

break;

case 'Q':

exit(0);

}

}

return 0;

}

/\* Вставка в очередь новой встречи. \*/

void enter(void)

{

char s[256], \*p;

do {

printf("Введите встречу %d: ", spos+1);

gets(s);

if(\*s==0) break; /\* запись не была произведена \*/

p = (char \*) malloc(strlen(s)+1);

if(!p) {

printf("Не хватает памяти.\n");

return;

}

strcpy(p, s);

if(\*s) qstore(p);

} while(\*s);

}

/\* Просмотр содержимого очереди. \*/

void review(void)

{

register int t;

for(t=rpos; t < spos; ++t)

printf("%d. %s\n", t+1, p[t]);

}

/\* Удаление встречи из очереди. \*/

void delete\_ap(void)

{

char \*p;

if((p=qretrieve())==NULL) return;

printf("%s\n", p);

}

/\* Вставка встречи. \*/

void qstore(char \*q)

{

if(spos==MAX) {

printf("List Full\n");

return;

}

p[spos] = q;

spos++;

}

/\* Извлечение встречи. \*/

char \*qretrieve(void)

{

if(rpos==spos) {

printf("Встречь больше нет.\n");

return NULL;

}

rpos++;

return p[rpos-1];

}