При изучении предыдущего примера программы планирования встреч, вероятно, вам в голову мог прийти следующий способ ее улучшения: при достижении конца массива, в котором хранится очередь, можно не останавливать программу, а устанавливать индексы вставки (spos) и извлечения (rpos) так, чтобы они указывали на начало массива. Это позволит помещать в очередь любое количество элементов при условии их своевременного извлечения. Такая реализация очереди называется *циклической очередью*, поскольку массив используется так, будто он представляет собой не линейный список, а кольцо.

Чтобы организовать в программе-планировщике циклическую очередь, функции qstore() и qretrieve() необходимо переписать следующим образом:

void qstore(char \*q)

{

/\* Очередь переполняется, когда spos на единицу

меньше rpos, либо когда spos указывает

на конец массива, а rpos - на начало.

\*/

if(spos+1==rpos || (spos+1==MAX && !rpos)) {

printf("Список полон\n");

return;

}

p[spos] = q;

spos++;

if(spos==MAX) spos = 0; /\* установить на начало \*/

}

char \*qretrieve(void)

{

if(rpos==MAX) rpos = 0; /\* установить на начало \*/

if(rpos==spos) {

printf("Встреч больше нет.\n");

return NULL;

}

rpos++;

return p[rpos-1];

}

В данной версии программы очередь переполняется, когда индекс записи находится непосредственно перед индексом извлечения; в противном случае еще есть место для вставки события. Очередь пуста, когда rpos равняется spos.

Вероятно, чаще всего циклические очереди применяются в операционных системах для хранения информации, учитываемой и записываемой в дисковые файлы или на консоль. Циклические очереди также используются в программах обработки реального времени, которые должны продолжать обрабатывать информацию, буферизируя при этом запросы на ввод/вывод. Многие текстовые процессоры используют этот прием во время переформатирования абзаца или выравнивания строки. Вводимый текст не отображается на экране до окончания процесса. Для этого прикладная программа должна проверять нажатие клавиш во время выполнения другой задачи. Если какая-либо клавиша была нажата, введенный символ быстро помещается в очередь, и процесс продолжается. После его завершения символы извлекаются из очереди.

Чтобы ощутить на практике данное применение циклических очередей, давайте рассмотрим простую программу, состоящую из двух процессов. Первый процесс в программе выводит на экран числа от 1 до 32 000. Второй процесс помещает символы в очередь по мере их ввода, не отображая их на экране, пока пользователь не нажмет <Enter>. Вводимые символы не отображаются, поскольку первому процессу дан приоритет в отношении вывода на экран. После нажатия <Enter> символы из очереди извлекаются и печатаются.

Чтобы программа работала, как описано выше, в ней необходимо использовать две функции, не определенные в стандартном языке С: \_kbhit() и \_getch(). Функция \_kbhit() возвращает значение ИСТИНА, если на клавиатуре была нажата клавиша; в противном случае она возвращает ЛОЖЬ. Функция \_getch() считывает введенный символ, но не дублирует его на экране. В стандарте языка С не предусмотрены функции для проверки состояния клавиатуры или считывания символов без отображения на экране, поскольку эти функции зависят от операционной системы. Тем не менее, в большинстве библиотек компиляторов есть функции, выполняющие данные задачи. Приведенная здесь небольшая программа компилируется с помощью компилятора Microsoft.

/\* Пример циклической очереди в качестве буфера клавиатуры. \*/

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#define MAX 80

char buf[MAX+1];

int spos = 0;

int rpos = 0;

void qstore(char q);

char qretrieve(void);

int main(void)

{

register char ch;

int t;

buf[80] = '\0';

/\* Принимать вводимые символы до нажатия <Enter>. \*/

for(ch=' ',t=0; t<32000 && ch!='\r'; ++t) {

if(\_kbhit()) {

ch = \_getch();

qstore(ch);

}

printf("%d ", t);

if(ch == '\r') {

/\* Вывести на экран содержимое буфера клавиатуры

и освободить буфер. \*/

printf("\n");

while((ch=qretrieve()) != '\0') printf("%c", ch);

printf("\n");

}

}

return 0;

}

/\* Занесение символа в очередь. \*/

void qstore(char q)

{

if(spos+1==rpos || (spos+1==MAX && !rpos)) {

printf("Список полон\n");

return;

}

buf[spos] = q;

spos++;

if(spos==MAX) spos = 0; /\* установить на начало \*/

}

/\* Получение символа из очереди. \*/

char qretrieve(void)

{

if(rpos==MAX) rpos = 0; /\* установить на начало \*/

if(rpos==spos) return '\0';

rpos++;

return buf[rpos-1];

}