1.4. ЧЕРГИ

Черги – це така лінійна структура, в якій включення елемента здійснюється через один кінець списку, який називається хвостом, а виключення через другий, який називається головою. Вони ще

називаються списками FIFO (First Input First Output), тобто елемент, який прийшов першим, буде й оброблятися першим (рис. 1.2).

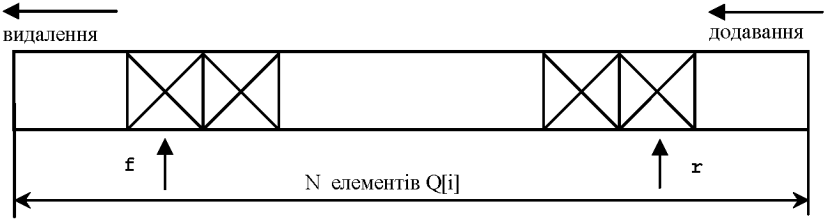

Рис. 1.2. Графічне представлення черги

Черга, на відміну від стека, має два вказівника:

r – вказівник хвоста черги;

f – вказівник голови черги.

Наведемо основні алгоритми роботи черги за умови, що для створення черги виділена пам'ять для розміщення n елементів (обсяг черги). При цьому мається на увазі, що початкові значення вказівників f та r нульові й використовується та ж нотація, що і в алгоритмах роботи зі стеком.

Алгоритм додавання елемента в чергу (QINSERT)

1) `if r>=n then <"повідомлення"; exit> {перевірка, чи не заповнена черга}`

2) `r:=r+1 {збільшення вказівника хвоста черги на одиницю}`

3) `Q[r] ← y {додавання елемента в чергу}`

4) `if f=0 then f:=1 {перше встановлення вказівника голови при першому включенні елемента до пустої черги}`

Алгоритм видалення елемента із черги (QDELETE)

1) `if f=0 then <"повідомлення"; exit> {перевірка, чи не пуста черга}`

2) `y ← Q[f] {видалення елемента із черги}`

3) `if f=r then (f:=0; r:=0) {якщо "хвіст наздогнав голову", тобто черга стала порожньою, то їх вказівники приймають початкові (нульові) значення}`

4) `f:=f+1 {збільшення вказівника голови черги на одиницю}`

На рис. 1.3 наведено приклад роботи черги. При цьому додавання елемента позначається символом «↑», а видалення елемента – символом «↓».

Рис. 1.3. Приклад роботи звичайної черги

Як бачимо, пам'ять при такій організації черги використовується не дуже раціонально: якщо в черзі є хоча б один елемент, то операцій додавання елементів можна виконати стільки, скільки пам'яті виділено під створення черги не дивлячись на те, що є ще вільні комірки, тобто вона не дозволяє по декілька разів використовувати комірки пам'яті, якщо до цього не було повного вивантаження пам'яті. Більш раціонально використовує пам'ять так звана *кругова* черга, у якій комірки пам'ять `Q[1],Q[2],...,Q[n-1],Q[n]` розташовані циклічно – за коміркою `Q[n]` слідує комірка `Q[1]` і коли комірка `Q[1]` звільниться (елемент видаляється із черги), то в ситуації, коли в черзі заповнена й комірка `Q[n]`, наступний елемент, що додається, може зайняти місце `Q[1]`.

Наведемо основні алгоритми роботи кругової черги за умови, що для створення черги виділена пам'ять для розміщення `n` елементів (обсяг черги). При цьому мається на увазі, що початкові значення вказівників `f` та `r` нульові.

Алгоритм додавання елемента в чергу (CQINSERT)

1) `if r=n then r:=1 else r:=r+1 {якщо черга заповнена, то перенести вказівник хвоста на початок, інакше він збільшується на 1}`

2) `if f=r then <"Повідомлення"; exit> {збільшення вказівника хвоста черги на 1}`

3) `Q[r]:=y {додавання елемента в чергу}`

4) `if f=0 then f:=1 {перше встановлення вказівника голови при першому включенні елемента до пустої черги}`

Алгоритм видалення елемента із черги (CQDELETE)

```
1) if f=0  then <"Повідомлення"; exit> {перевірка,
чи не пуста черга}
2) y ← Q[f] {видалення елемента із черги}
3) if f=r then (f:=0; r:=0; exit) {якщо "хвіст
наздогнав голову", тобто черга стала порожньою, то
їх вказівники приймають початкові (нульові) значення}
4) f:=f+1 {збільшення вказівника голови черги на
одиницю}
```

Приклад роботи кругової черги наведено на рис. 1.4. Тут обсяг черги також складає чотири елемента.

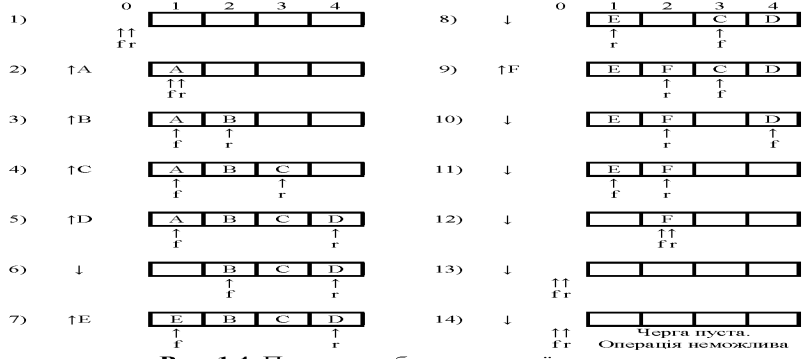

Рис. 1.4. Приклад роботи кругової черги

Черги, як і стеки, широко використовуються в програмних системах (ОС), наприклад, в операційних системах, оскільки усі сучасні ОС є багатозадачними, а задачі можуть одночасно звертатися до одних і

тих самих ресурсів. Управління такою чергою є функцією планувальника завдань ОС. Деякі з алгоритмів черг планувальників завдань ОС буде розглянуто нижче. Черги і стеки є також обов'язковими програмними структурними одиницями багатьох систем моделювання, наприклад, мови GPSS.