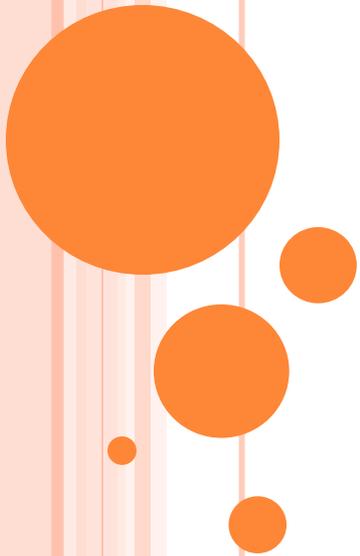


СВЯЗНЫЕ СПИСКИ.

Очередь



ОЧЕРЕДЬ

Очередь –линейный список, организованный по принципу первый пришел – последний ушел (FIFO – first in first out).

Бывает однонаправленная и двунаправленная

Голова (вершина) очереди – самый верхний элемент

Хвост (конец) очереди – самый нижний (последний) элемент

Добавление новых элементов допустимо только с хвоста очереди.

Удаление существующих элементов допустимо только с головы очереди



ОЧЕРЕДЬ

Однонаправленная

```
struct FIFO {  
    int data;  
    struct FIFO *next;  
};
```

Двухнаправленная

```
struct FIFO {  
    int data;  
    struct FIFO *next;  
    struct FIFO *prev;  
};
```

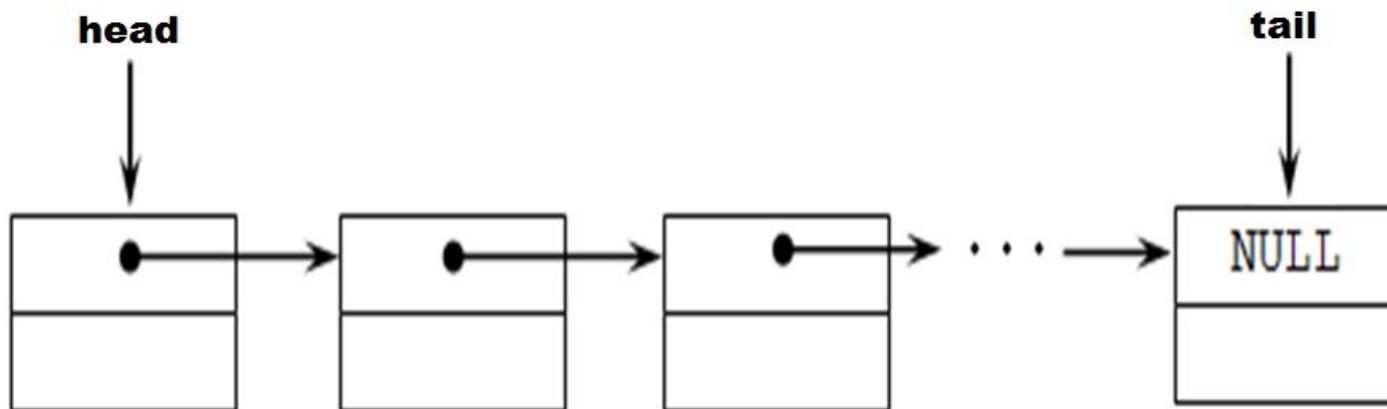


ОЧЕРЕДЬ

Добавление

Удаление

Неразрушающий просмотр



ДОБАВЛЕНИЕ

Создать временный указатель

Выделить место

Записать данные в информационное поле

Указатель next сделать NULL

Проверить хвост

Если NULL

 Хвост=временный указатель

Иначе

 Хвост->Next = временный указатель

 Хвост = временный указатель

Вернуть голову



ДОБАВЛЕНИЕ

```
struct FIFO * create(struct FIFO *tail, int x)
{
    struct FIFO *n;
    n = (struct FIFO *)malloc(sizeof(struct FIFO));
    n->next = NULL;
    n->data = x;
    if (tail == NULL)
    {
        tail= n;
    }
    else
    {
        tail->next=n;
        tail=n;
    }
    return tail;
}
```



ДОБАВЛЕНИЕ

```
void create(struct FIFO **head, struct FIFO **tail, int x)
{
    struct FIFO *n;
    n = (struct FIFO *)malloc(sizeof(struct FIFO));
    n->next = NULL;
    n->data = x;
    if (*head == NULL)
    {
        *head = n;
        *tail = n;
    }
    else
    {
        (*tail)->next=n;
        *tail=n;
    }
}
```



УДАЛЕНИЕ

Если голова не NULL

Голова = голова ->next;

Вернуть голову

Иначе

Вернуть NULL



УДАЛЕНИЕ

```
struct FIFO *pop(struct FIFO * head)
{
if (head != NULL)
{
    head = head->next;
    return head;
}
else return NULL;
}
```



УДАЛЕНИЕ

```
void pop(struct FIFO **head, struct FIFO ** tail)
{
    if (*head != NULL)
    {
        *head = (*head)->next;
    }
    else
    {
        *tail=NULL;
    }
}
```



ПРОСМОТР

Создать временный указатель

Временный указатель = голова

Пока временный указатель не равен NULL

Вывод на экран информационного поля временного указателя

Передвинуть временный указатель на следующий элемент



ПРОСМОТР

```
struct FIFO *p = head;
while (p != NULL)
{
    printf("%d -->", p->data);
    p = p->next;
}
```



ПРИМЕР 1

Создать очередь из натуральных чисел до N . Вывести на экран.



ПРИМЕР 2

Создать очередь из чисел введенных пользователем. Вывести на экран.

Посчитать сумму элементов



ПРИМЕР 3

Создать очередь из чисел введенных пользователем. Вывести на экран. Удалить первые K чисел



ПРИМЕР 4

Найти максимальное значение в очереди. Удалить все элементы до/после него



ОЧЕРЕДЬ ДВУНАПРАВЛЕННАЯ

```
struct FIFO {  
    int data;  
    struct FIFO *next;  
    struct FIFO *prev;  
  
};
```



ОЧЕРЕДЬ ДВУНАПРАВЛЕННАЯ

Добавление

Удаление

Неразрушающий просмотр



ДОБАВЛЕНИЕ

Создать временный указатель tmp

Выделить место

Записать данные в информационное поле

Указатель tmp->next сделать NULL

Указатель tmp->prev сделать NULL

Проверить хвост

Если NULL

 Хвост=временный указатель

Иначе

 tmp->prev = хвост

 Хвост->Next = tmp

 хвост = tmp

Вернуть хвост



ДОБАВЛЕНИЕ

```
struct FIFO * create(struct FIFO *tail, int x)
{
    struct FIFO *n;
    n = (struct FIFO *)malloc(sizeof(struct FIFO));
    n->next = NULL;
    n->prev = NULL;
    n->data = x;
    if (tail == NULL)
    {
        tail= n;
    }
    else
    {
        n->prev=tail;
        tail->next=n;
        tail=n;
    }
    return tail;
}
```



ДОБАВЛЕНИЕ

```
void create(struct FIFO **head, struct FIFO **tail, int x)
{
    struct FIFO *n;
    n = (struct FIFO *)malloc(sizeof(struct FIFO));
    n->next = NULL;
    n->prev = NULL;
    n->data = x;
    if (*head == NULL)
    {
        *head = n;
        *tail = n;
    }
    else
    {
        n->prev=(*tail);
        (*tail)->next=n;
        *tail=n;
    }
}
```



УДАЛЕНИЕ

Удаление только с головы!

Если голова не NULL

голова = голова ->next;

голова->prev =NULL;

Вернуть голову

Иначе

Вернуть NULL



УДАЛЕНИЕ

```
struct FIFO *pop(struct FIFO * head)
{
if (head != NULL)
{
    head = head->next;
    head->prev=NULL;
    return head;
}
else return NULL;
}
```



УДАЛЕНИЕ

```
void pop(struct FIFO **head, struct FIFO ** tail)
{
    if (*head != NULL)
    {
        *head = (*head) ->next;
        (*head) ->prev=NULL;
    }
    else
    {
        *tail=NULL;
    }
}
```



ПРОСМОТР

С головы:

Создать временный указатель

Временный указатель = голова

Пока временный указатель не равен NULL

Вывод на экран информационного поля временного указателя

Передвинуть временный указатель на следующий элемент

С хвоста:

Создать временный указатель

Временный указатель = хвост

Пока временный указатель не равен NULL

Вывод на экран информационного поля временного указателя

Передвинуть временный указатель на предыдущий элемент



ПРОСМОТР

```
struct FIFO *p = head;
while (p != NULL)
{
    printf("%d -->", p->data);
    p = p->next;
}
```

```
struct FIFO *p = tail;
while (p != NULL)
{
    printf("%d -->", p->data);
    p = p->prev;
}
```



ПРИМЕР 5

Создать двунаправленную очередь из натуральных чисел до N . Вывести на экран.



ПРИМЕР 6

Первая очередь упорядочена по возрастанию, вторая по убыванию. Создать третью упорядоченную по неубыванию.

