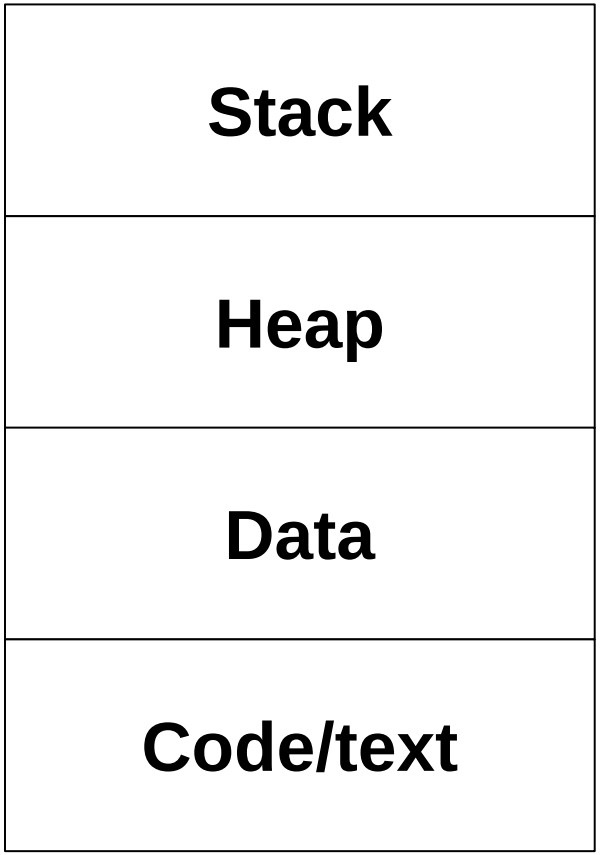


Stack

Stack is a linear data structure Which follows LIFO or FILO operation.

LIFO --> Last In First Out
FILO --> Fist In Last Out.

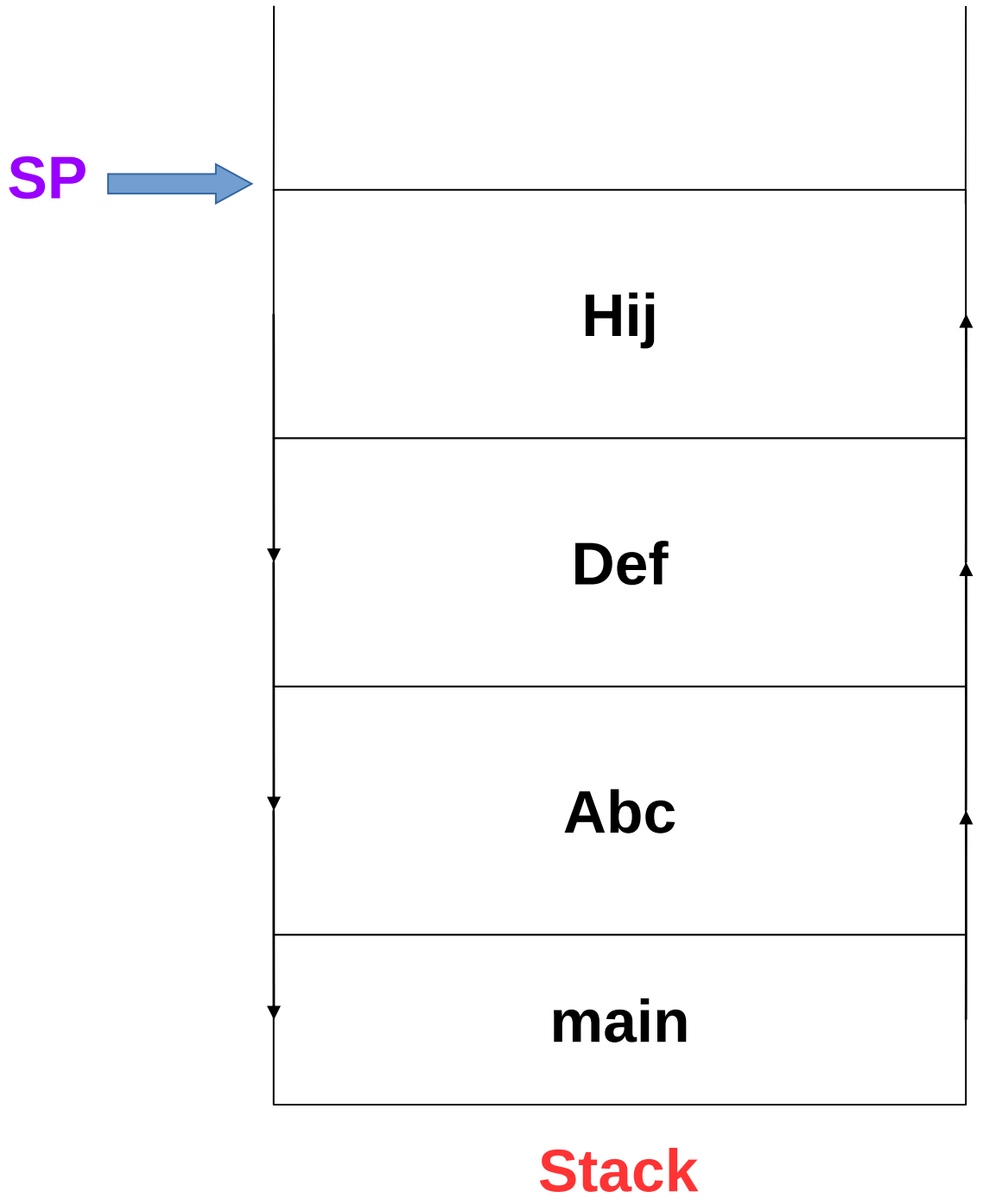
Stack containes 2 operations.

1. push : to insert the arguments.
2. pop : to delete the arguments.

```
Ex : main()
  {
     Abc();
  }

     Void Abc()
     {
     ......
     Def();
     }
     Void Def()
     {
     ......
     Hij();
     }

     Void Hij()
     {
     ......
     ......
     }
```



Stack

Stack makes the program execution in a proper manner.

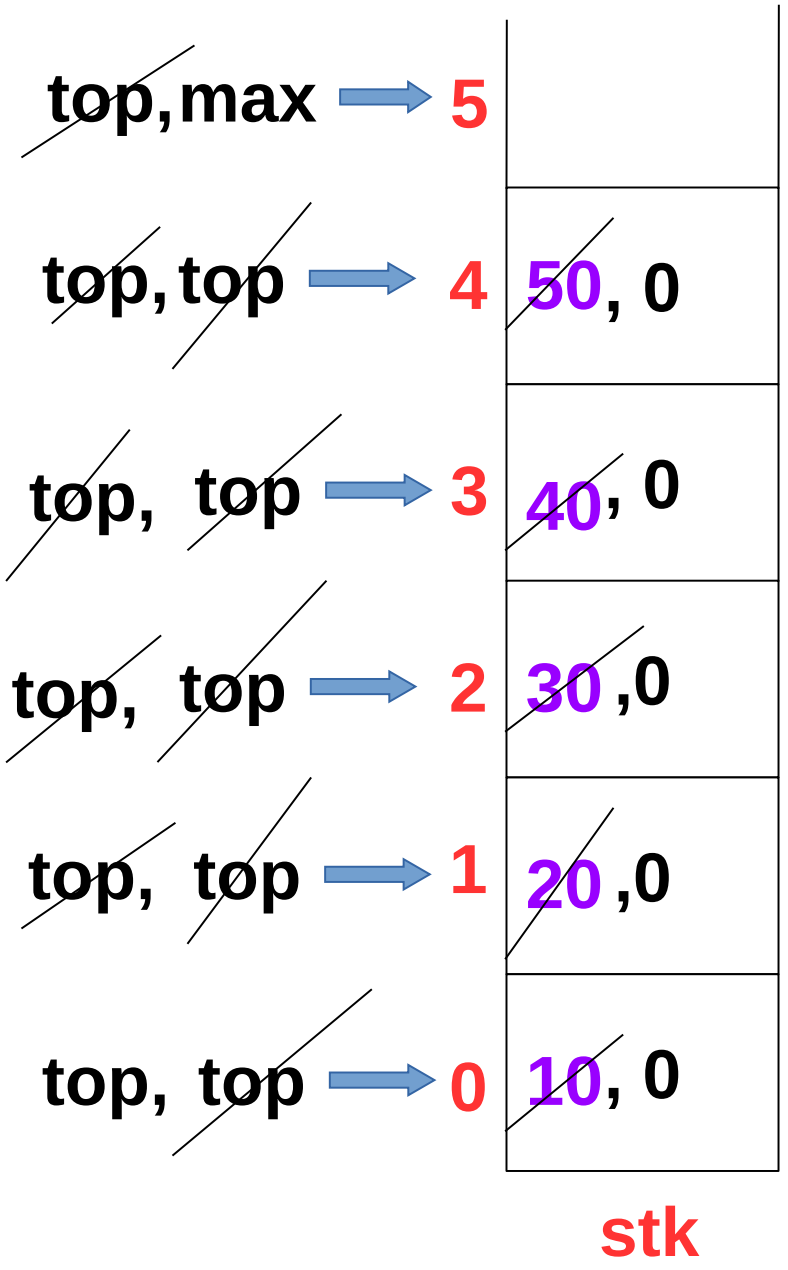
```
#define max 5;
int stk[max];
int top = 0;
```

push

```
if(top == max) {
printf("stack is overflow\n");
return;
}
else
stk[top++] = data;
```

pop

```
if(top == 0) {
printf("stack is underflow\n");
return;
}
else
stk[--top] = 0;
```

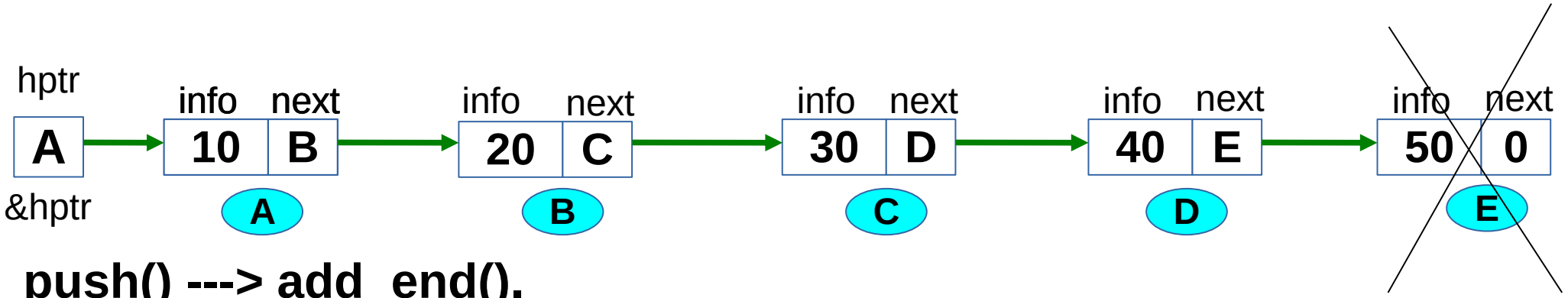


```
1 //WAP to implement menu based stack using arrays.
2 #include<stdio.h>
3 #define max 5
4 int stk[max];
5 int top = 0;
6 void push();
7 void pop();
8 void display();
9 int main()
10 {
11    int op;
12    while(1)
13    {
14        printf("Enter the option 1)push 2)pop 3)display 4)exit\n");
15        scanf("%d",&op);
16        switch(op)
17        {
18              case 1 : push(); break;
19              case 2 : pop(); break;
20              case 3 : display(); break;
21              case 4 : return 0;
22              default : printf("Invalid option...\n");
23                        return 0;
24        }
25    }
26 }
```

```
27 void push()
28 {
29        if(top == max) {
30        printf("stack is overflow...\n");
31        return;
32        }
33
34        int ele;
35        printf("Enter the element to push\n");
36        scanf("%d",&ele);
37        stk[top++] = ele;
38 }
39 void pop()
40 {
41        if(top == 0) {
42        printf("stack is underflow...\n");
43        return;
44        }
45
46        printf("%d is popped...\n",stk[top-1]);
47        stk[--top] = 0;
48 }
```

```
49 void display()
50 {
51        if(top == 0) {
52        printf("stack is empty...\n");
53        return;
54        }
55
56        int i;
57        for(i=0;i<top;i++)
58        printf("%d ",stk[i]);
59        printf("\n");
60 }
```

Stack implementation using linked list:



push() ---> add_end().
pop() ---> delete last node always.

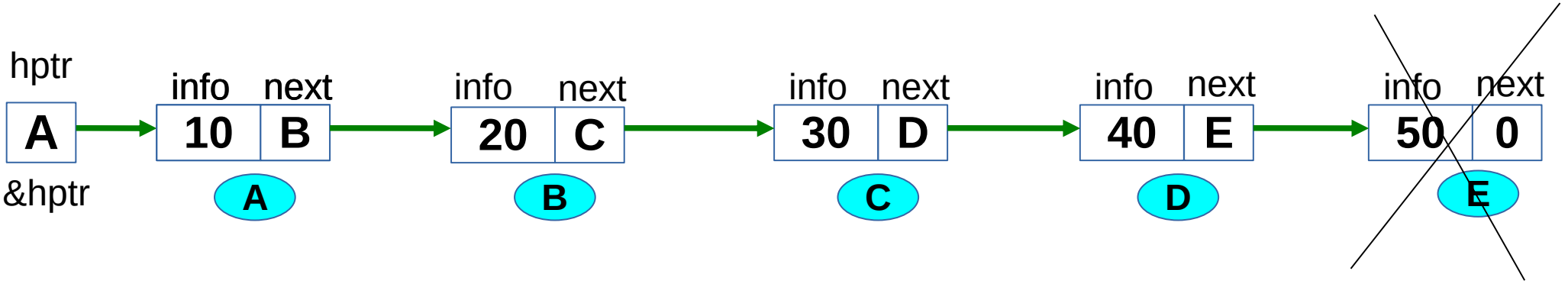
```
#define max 5
int node_count = 0; //global
```

push(&hptr); ⟶

```
void push(ST **ptr)
{
if(node_count == max) {
printf("Stack is overflow\n");
return;
}

//logic for add_end

node_count++;
}
```

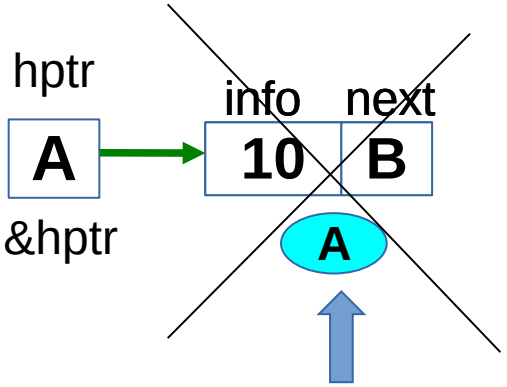


pop(&hptr) --->

```
void pop(ST **ptr)
{
ST *last = *ptr;
ST *prv;                          //delete last node

while(last->next != 0) {
prv = last;
last = last->next;
}

prv->next = last->next;
}
```



***ptr**

```
                                    //delete first node
pop(&hptr) --->  void pop(ST **ptr)
                 {
                 if((*ptr)->next == 0) {
                 free(*ptr);
                 *ptr = 0;
                 }
                  }
```

```
void pop(ST **ptr)
{
ST *prv, *last = *ptr;

if(node_count == 0)  {
printf("Stack is underflow...\n");
return;
}
while(last->next != 0) {
prv = last;
last = last->next;
}

if(last == *ptr)
*ptr = 0;
else
prv->next = last->next;
free(last);

node_count--;
}
```

Queue

Queue is a linear data structure and which follows FIFO or LILO

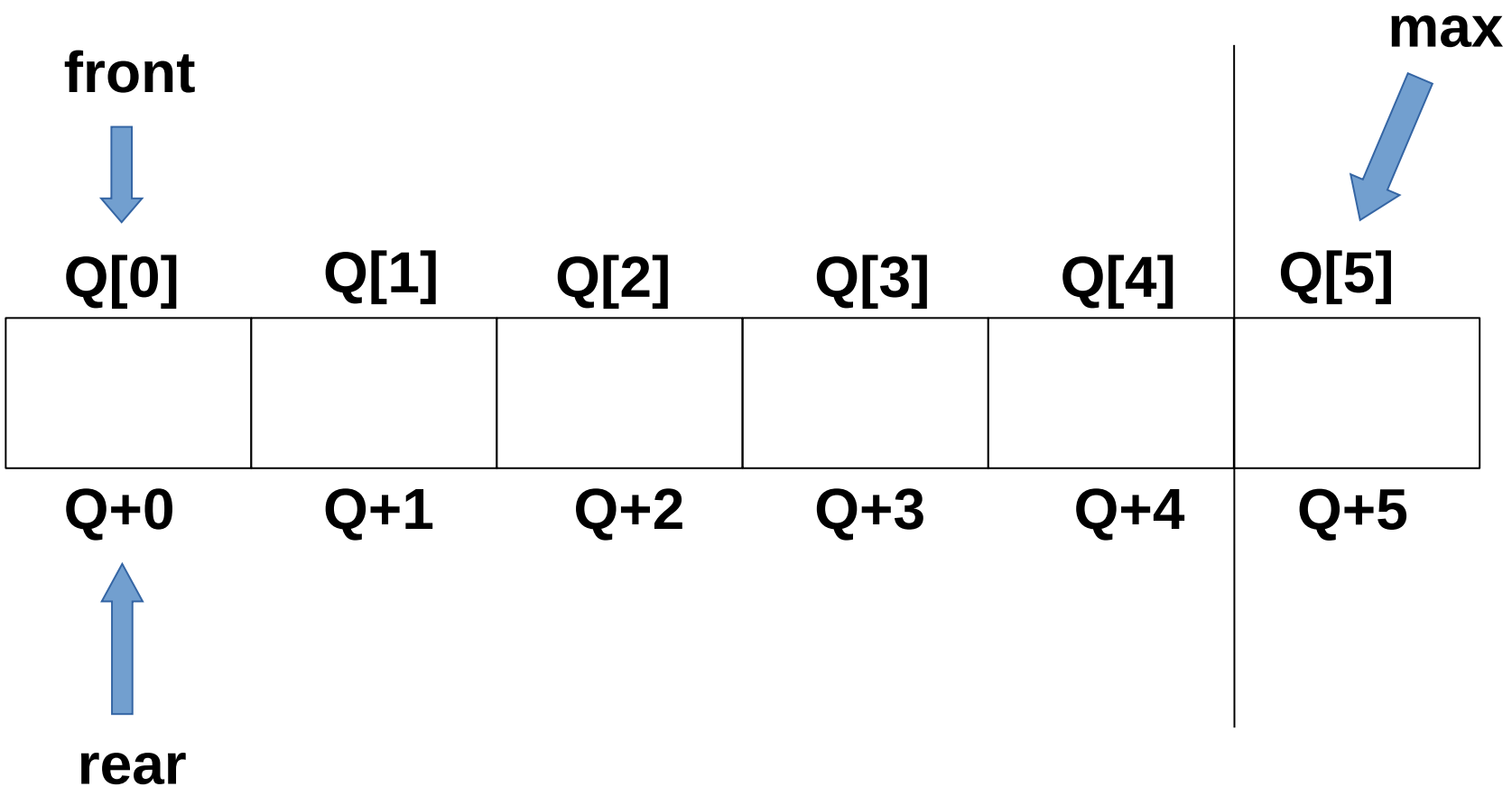
FIFO ---> First In First Out
LILO ---> Last In Last Out

To perform Queue operations there are 2 functions.

1) enqueue() --> to insert the data (rear)
2) dequeue() --> to delete the data (front)

Implimentation of Queue using Arrays

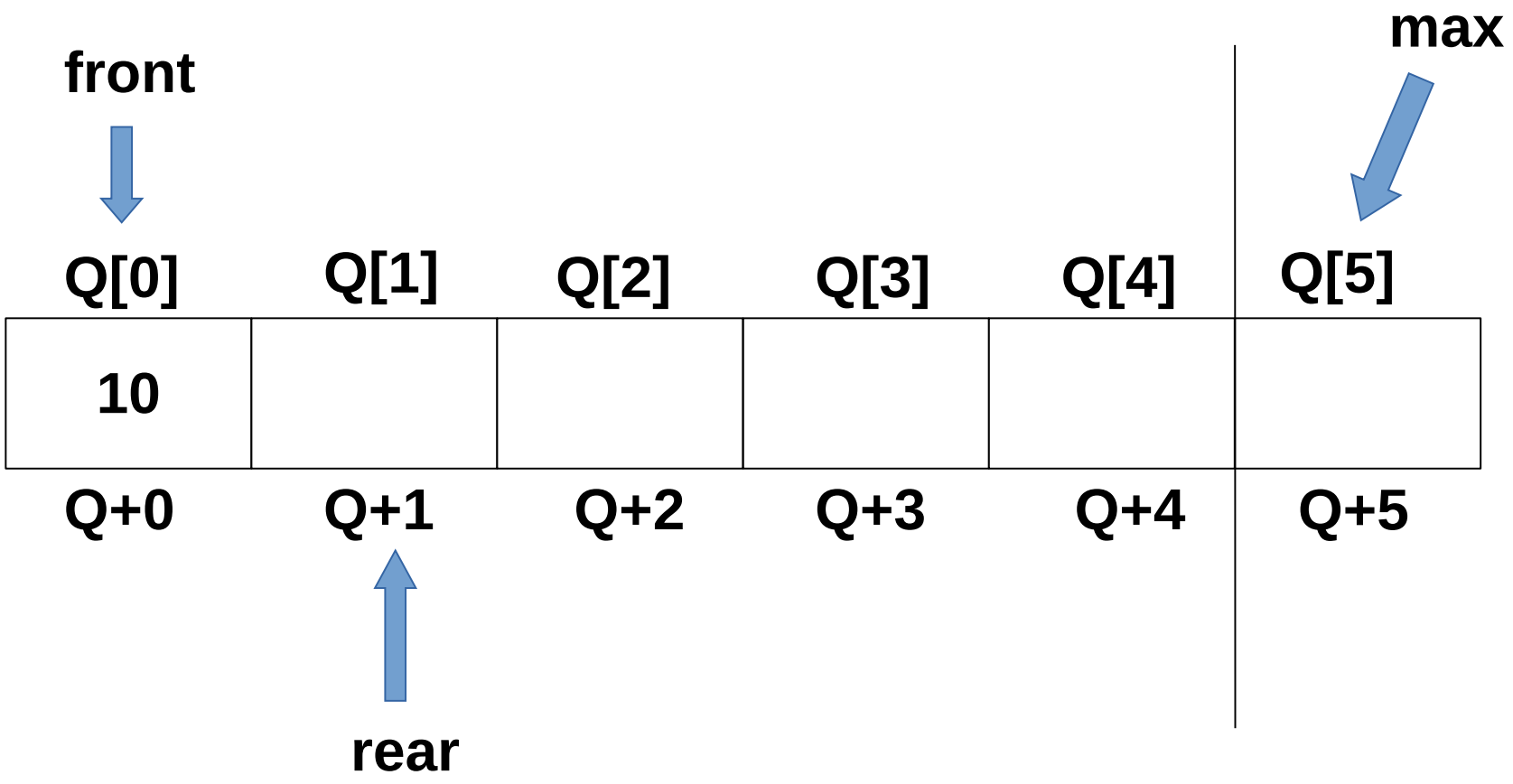
```
#define max 5
int Q[max];
```



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

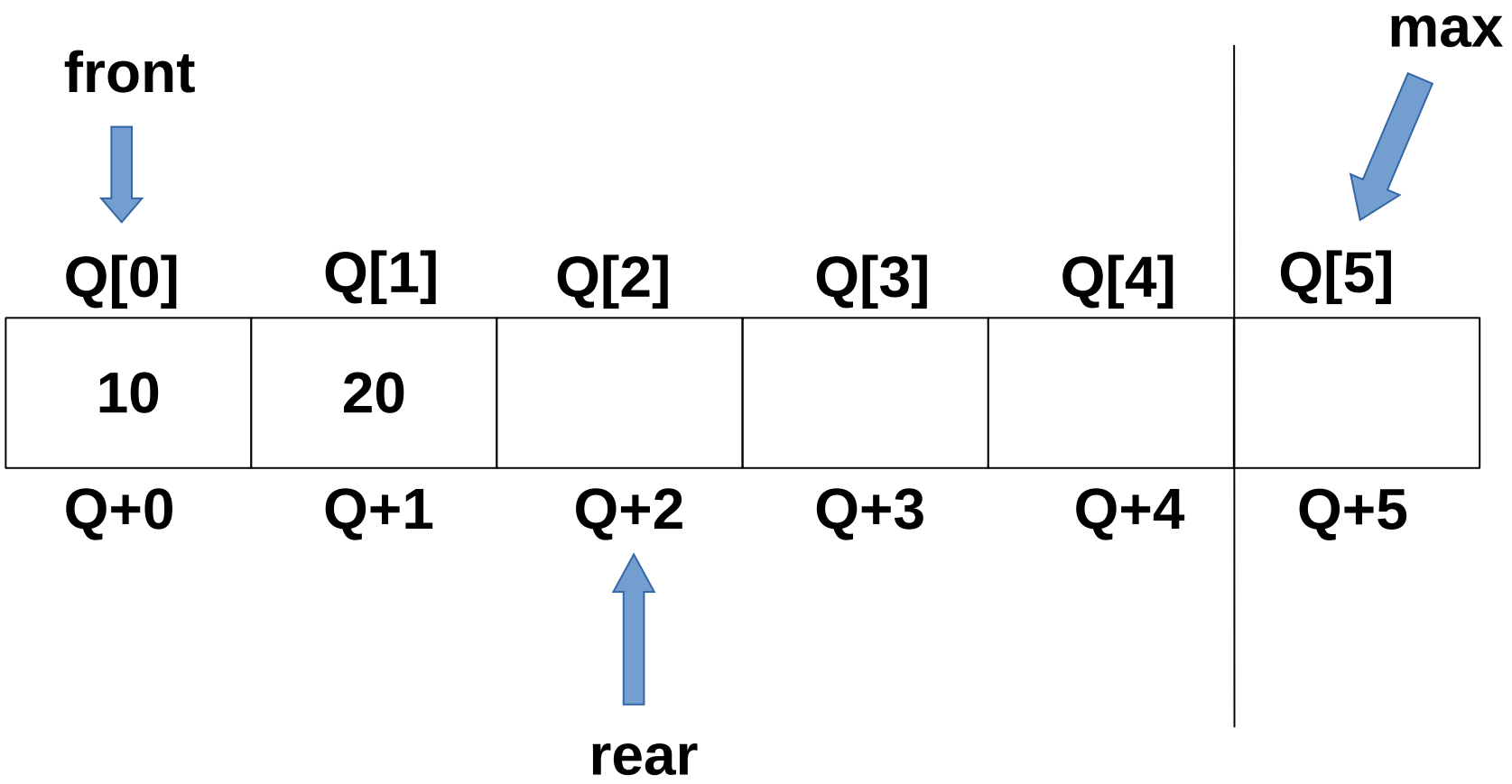
Insert value 10



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

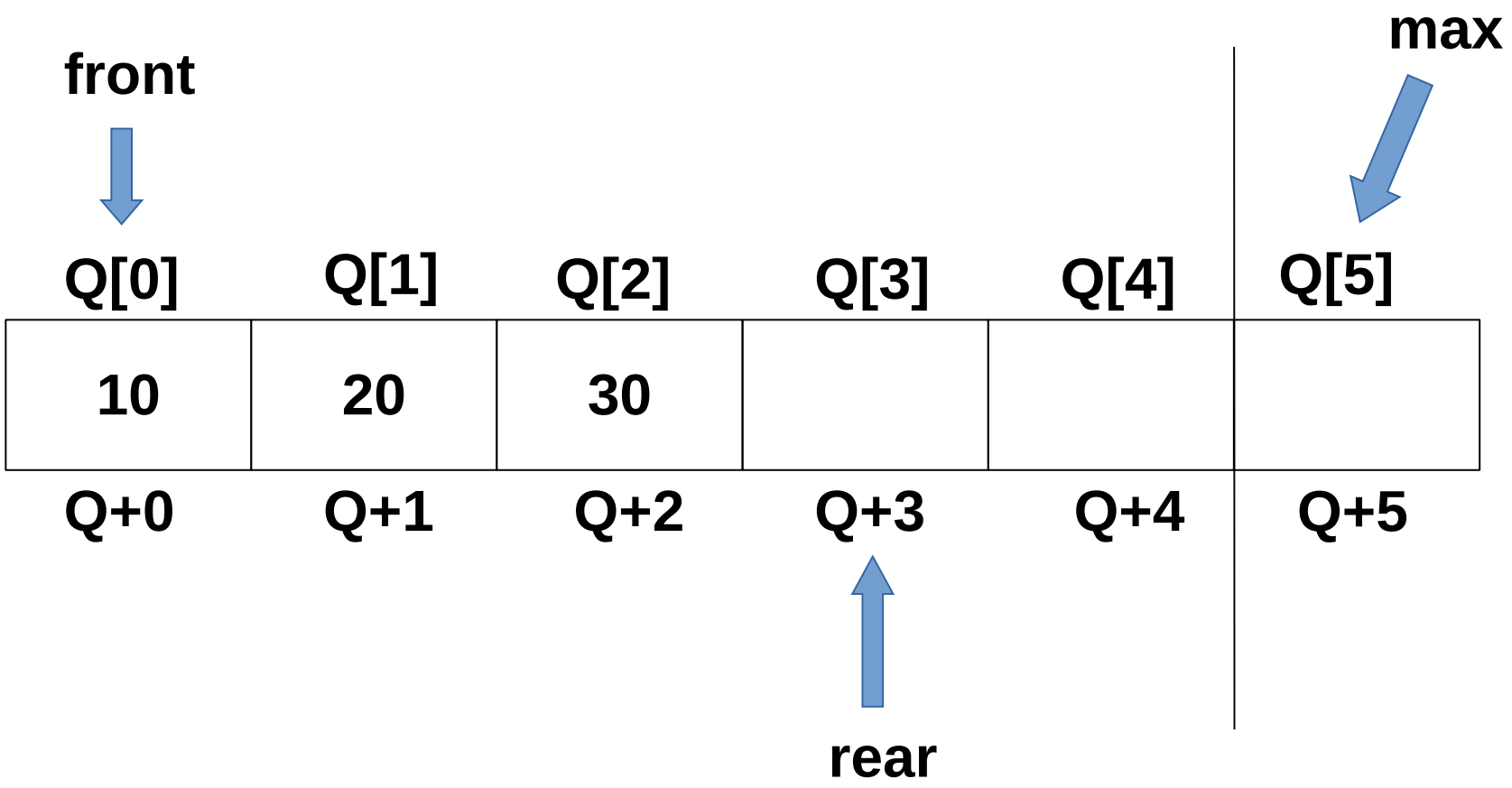
Insert value 20



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

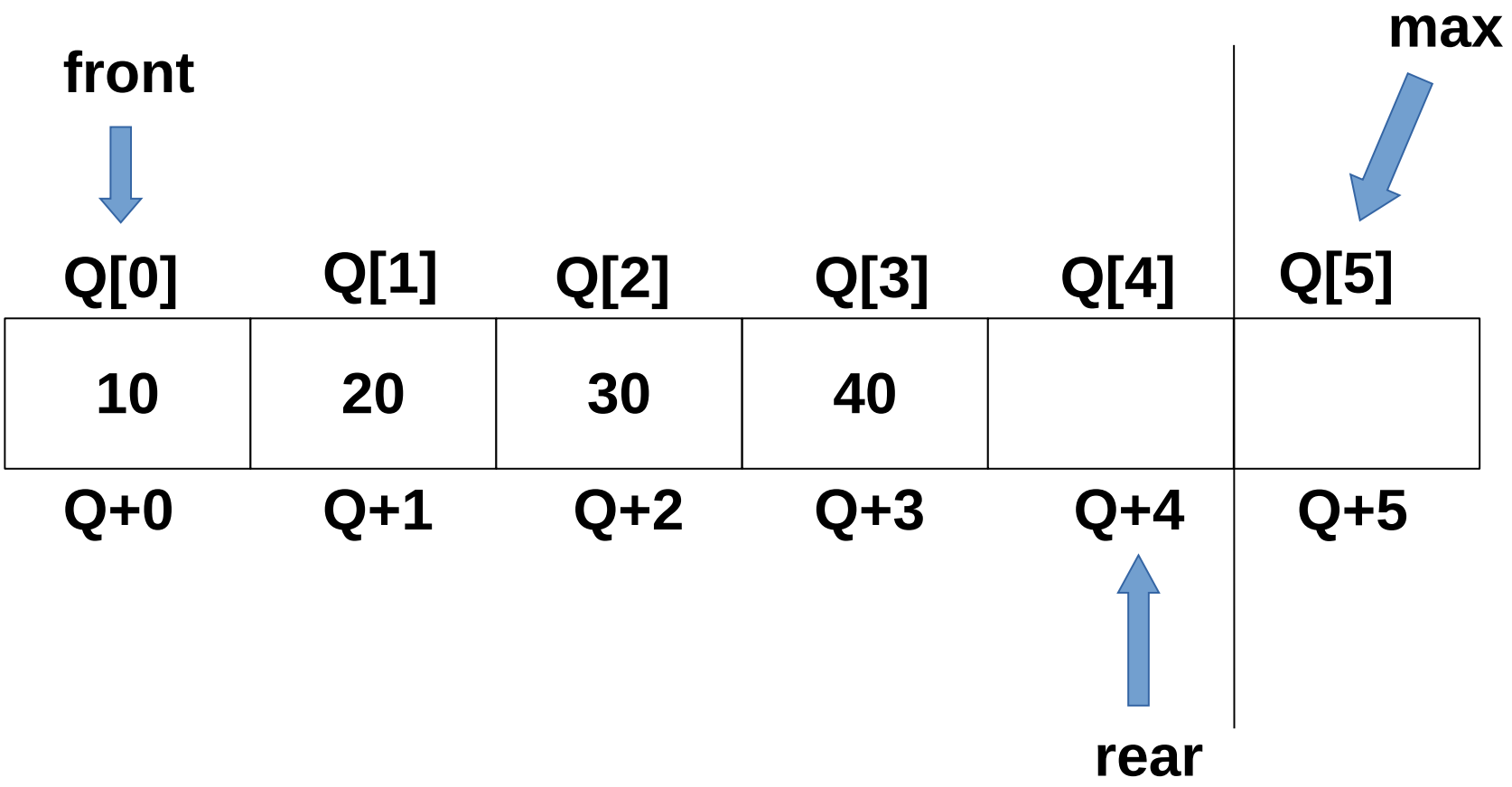
Insert value 30



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

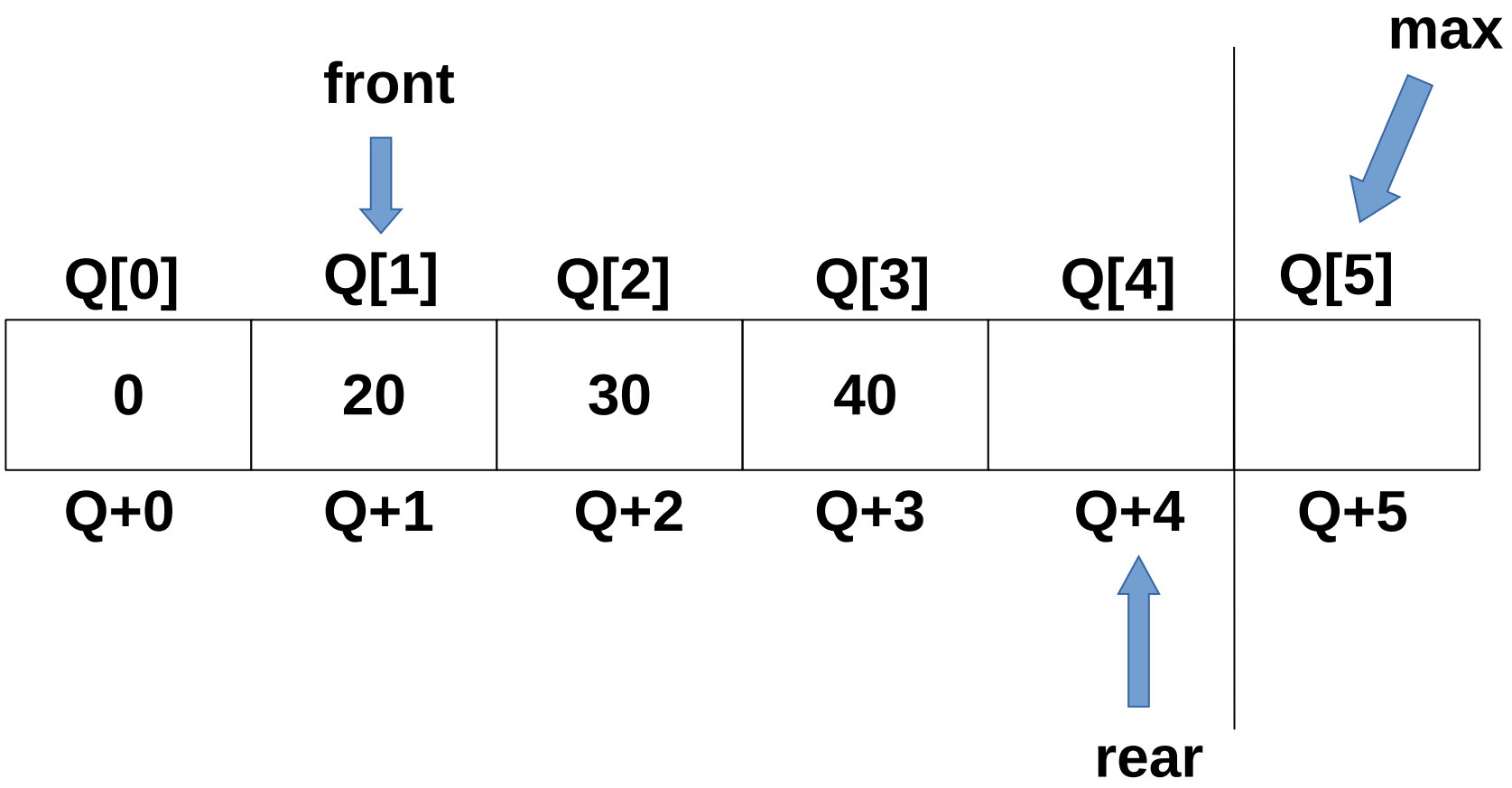
Insert value 40



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

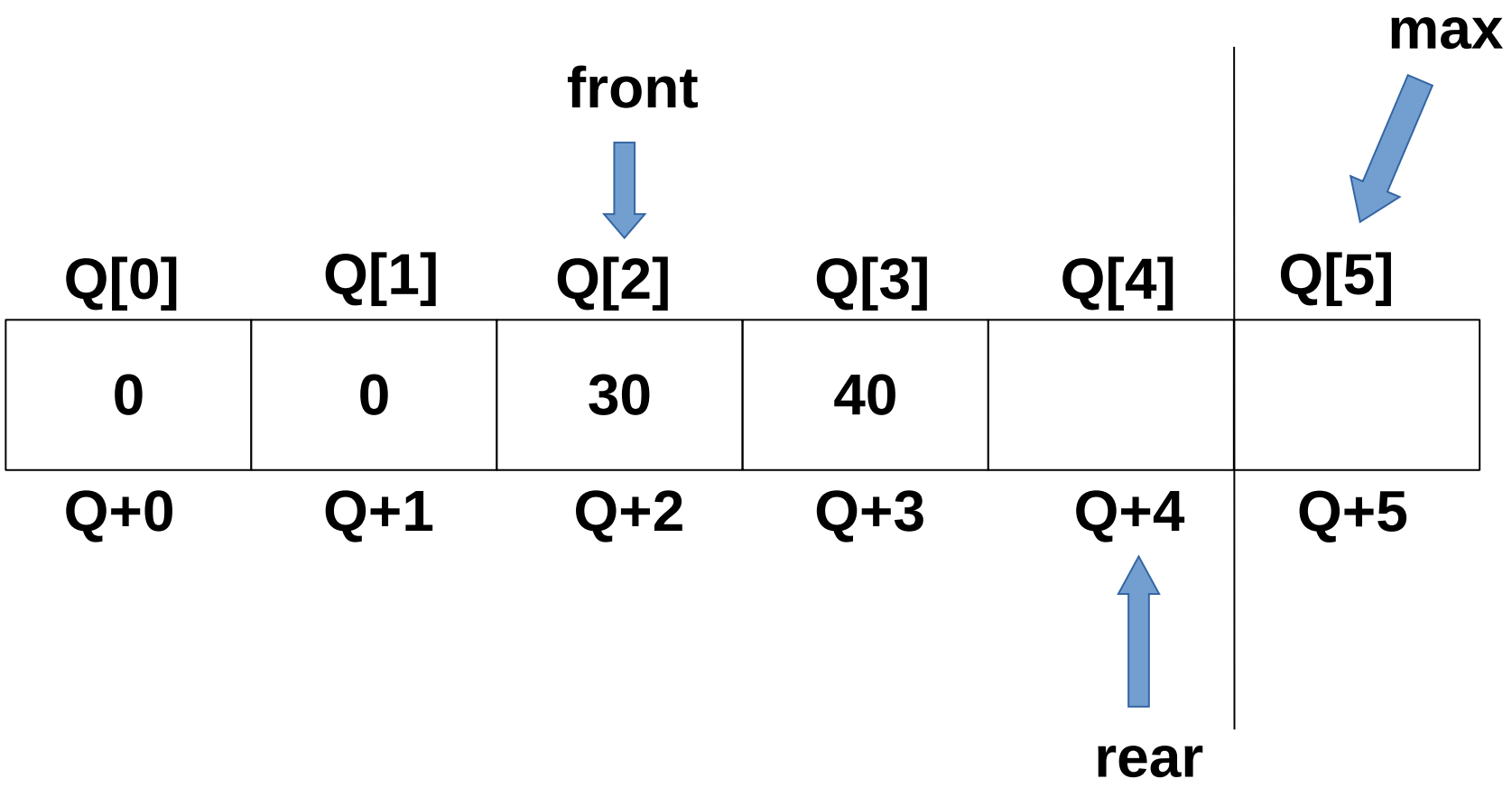
delete value 10



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

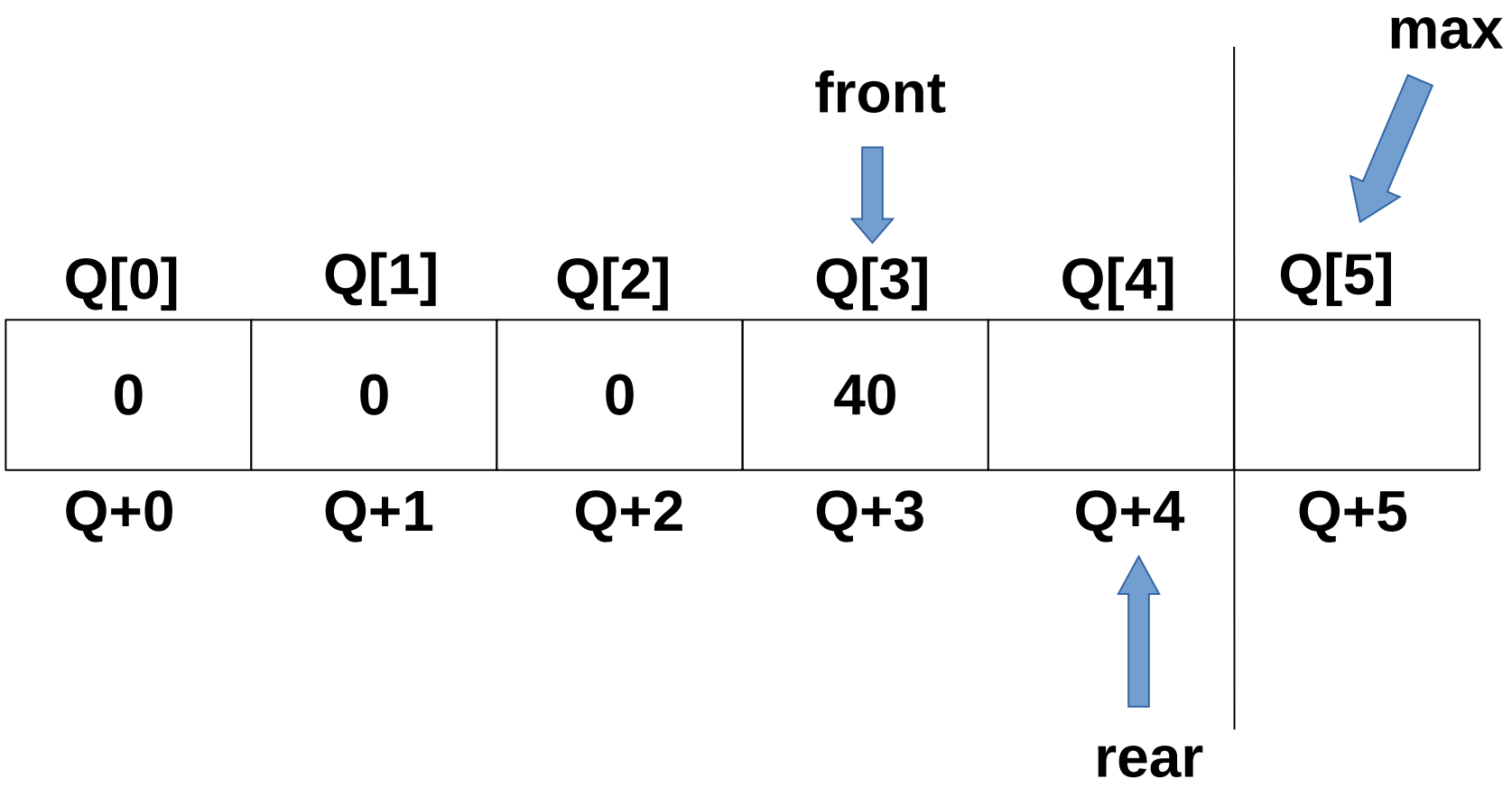
delete value 20



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

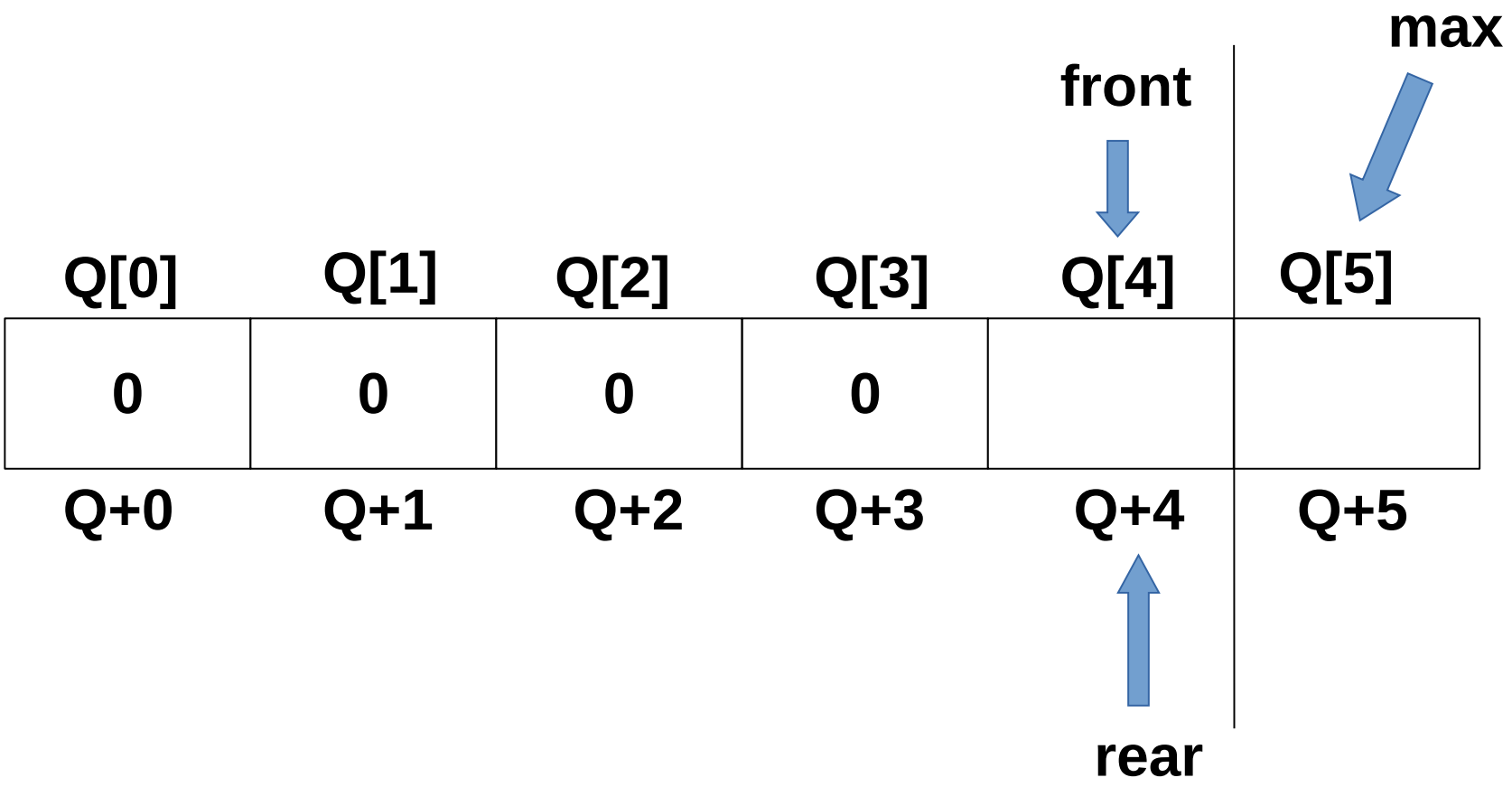
delete value 30



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

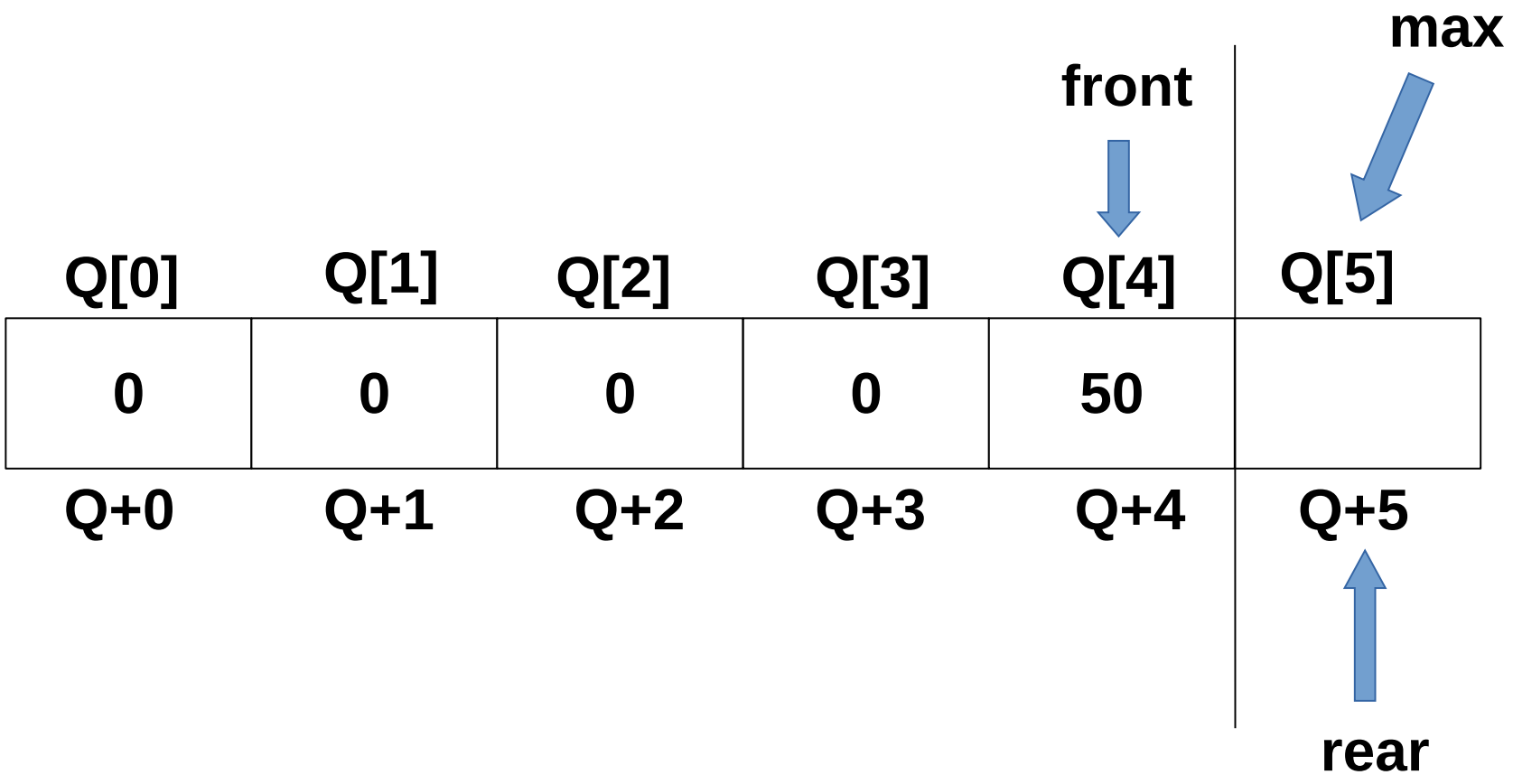
delete value 30



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```

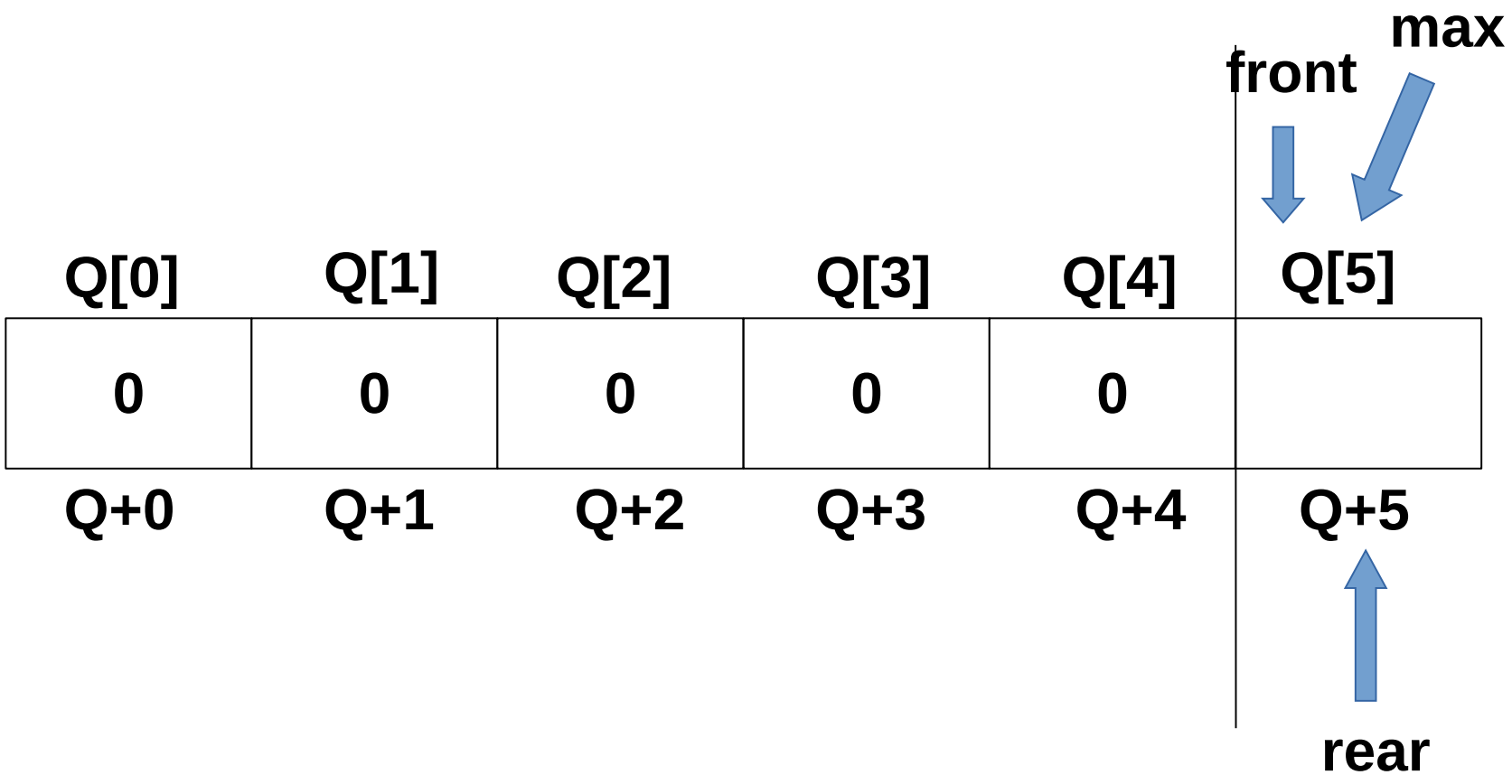
insert value 50



Implimentation of Queue using Arrays

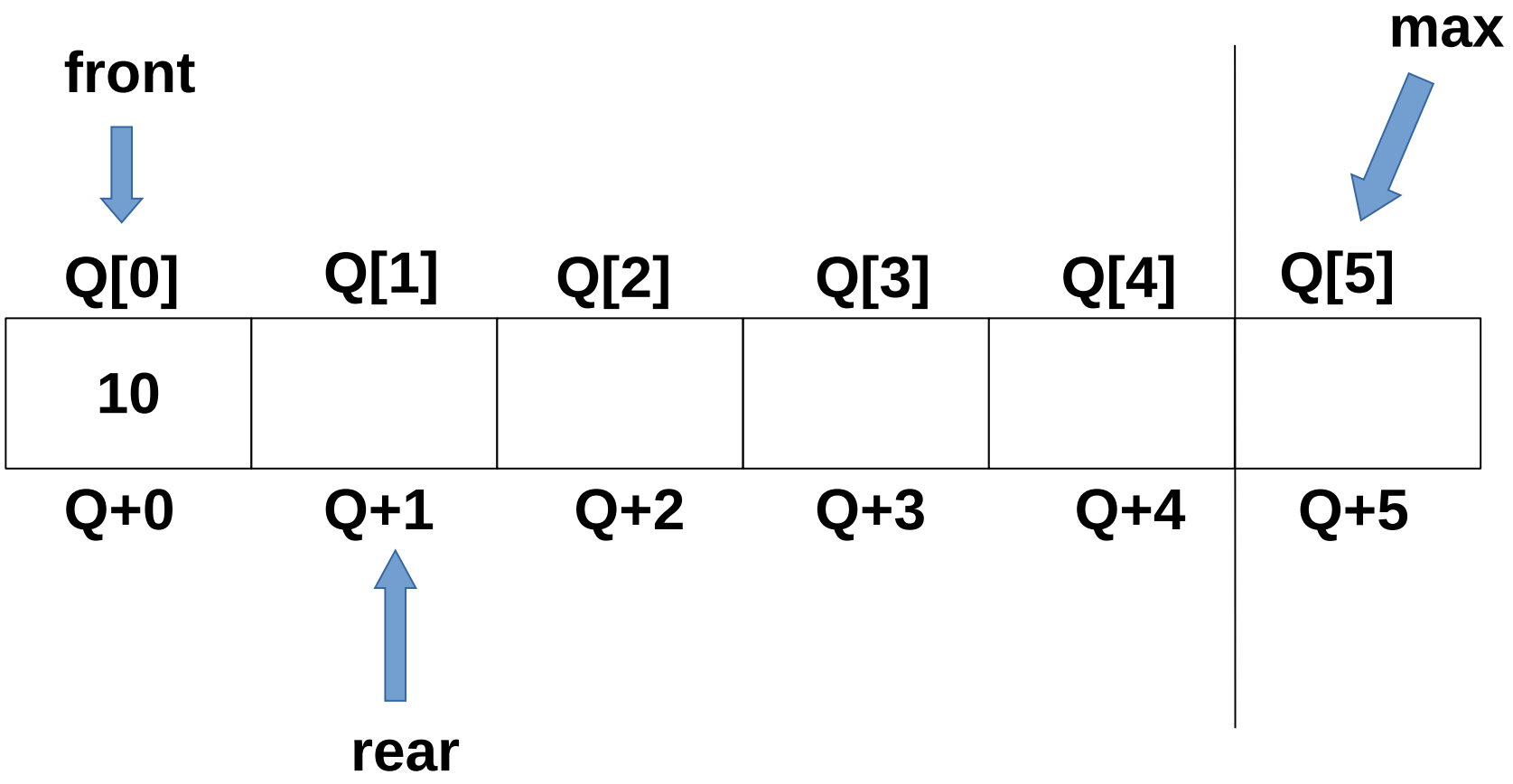
```
#define max 5
int Q[max];
```

insert value 50



Implimentation of Queue using Arrays

```
#define max 5
int Q[max];
```



```
//implementation of Queue using Arrays
#include<stdio.h>
#define MAX 5
int Q[MAX],front = 0,rear = 0;
void enqueue();
void dequeue();
void display();
int main()
{
        int op;
         while(1)
         {
                printf("Enter the option 1)insert 2)delete 3)display 4)exit\n");
                scanf("%d",&op);

                switch(op)
                {
                        case 1 : enqueue(); break;
                        case 2 : dequeue(); break;
                        case 3 : display(); break;
                        case 4 : printf("Q operation is Over!\n");return;
                        default: printf("Invalid option entered...\n"); break;
                }
         }
}
```

```
25 void enqueue()
26 {
27         int ele;
28
29         if(rear == MAX) {
30         printf("Q is Overflow...\n");
31         return;
32         }
33
34         printf("Enter the element to insert...\n");
35         scanf("%d",&ele);
36
37         Q[rear++] = ele;
38
39         printf("%d is inserted into Q...\n",ele);
40 }
```

```
41 void dequeue()
42 {
43         if((rear == 0)||(front==rear)) {
44         printf("Q is Underflow...\n");
45         return;
46         }
47
48         printf("%d is deleted from Q...\n",Q[front]);
49         Q[front++] = 0;
50 }
51 void display()
52 {
53         int i;
54         for(i=0;i<MAX;i++)
55         printf("%d ",Q[i]);
56         printf("\n");
57 }
```

```
//implementation of circular queue
 1 #include<stdio.h>
 2 #define MAX 5
 3 int CQ[MAX],front = 0,rear = 0,flag = 0;
 4 void insert();
 5 void delete();
 6 void display();
 7 int main()
 8 {
 9         int op;
10          while(1)
11          {
12                  printf("Enter the option 1)insert 2)delete 3)display 4)exit\n");
13                  scanf("%d",&op);
14
15                  switch(op)
16                  {
17                          case 1 : insert(); break;
18                          case 2 : delete(); break;
19                          case 3 : display(); break;
20                          case 4 : return;
21                          default: printf("Invalid option entered...\n"); break;
22                  }
23          }
24 }
```

```
25 void insert()
26 {
27         int num;
28
29         if(flag && rear >= front) {
30         printf("CQ is Overflow...\n");
31         return;
32         }
33
34         printf("Enter the data to insert...\n");
35         scanf("%d",&num);
36
37         CQ[rear++] = num;
38         printf("%d is inserted in CQ...\n",num);
39
40         if(rear == MAX) {
41         flag = !flag;
42         rear = 0;
43         }
44 }
```

```
45 void delete()
46 {
47
48        if( (!flag)&&(front >= rear) )  {
49        printf("CQ is Underflow...\n");
50        return;
51        }
52
53        printf("%d is deleted...\n",CQ[front]);
54        CQ[front++] = 0;
55
56        if(front == MAX) {
57        flag = !flag;
58        front = 0;
59        }
60 }
61 void display()
62 {
63        int i;
64        for(i=0;i<5;i++)
65        printf("%d ",CQ[i]);
66        printf("\n");
67 }
```