Queue and Tips on Programming Assignment 1

Roy Chan

CSC2100B Data Structures Tutorial 3

February 3, 2009

1 Queue

- Overview
- Implementation Using Array
- Implementation Using Linked List

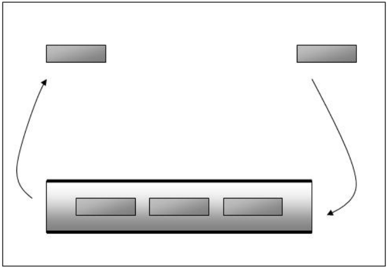
2 Tips on Programming Assignment 1

- Common Questions
- ex1_13
- ex1_14

Queue

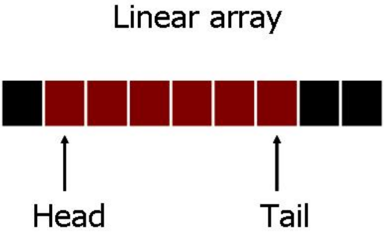
Queue Overview

- First In First Out (FIFO)
- Enqueue
- Dequeue



Queue Implementation

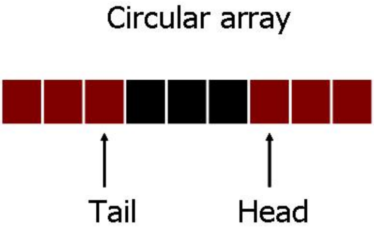
- A queue may be implemented using linked-list or array
- Implement a queue using array



Queue Implementation Using Array

- Implementing a queue using circular array

```
typedef struct {
  int *data;     //data is an array of int
  int head;
  int tail;
  int num;       //number of elements in queue
  int size;      //size of queue
}Queue;
```



Queue Implementation Using Array

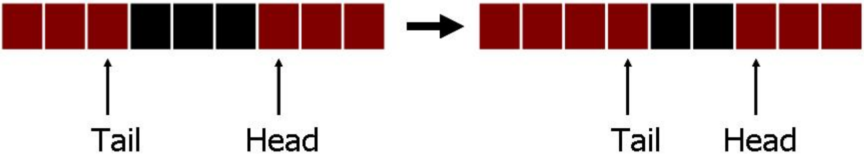
- createQueue

```c
//return 1 for success, 0 for fail
int createQueue(Queue *aqueue, int size)
{
  aqueue->data = (int*)malloc(sizeof(int)*size);
  if (aqueue->data == NULL)
    return 0;

  aqueue->head = 0;
  aqueue->tail = -1;
  aqueue->num = 0;
  aqueue->size = size;

  return 1;
}
```

Queue Implementation Using Array

- enqueue

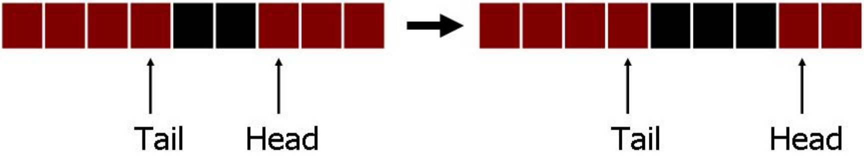
```c
void enqueue(Queue *aqueue, int adata)
{
  aqueue->tail = (aqueue->tail+1)%aqueue->size;
  aqueue->data[queue->tail] = adata;
  aqueue->num++;
}
```



Queue Implementation Using Array

- dequeue

```c
int dequeue(Queue *aqueue)
{
  int adata = aqueue->data[aqueue->head];
  aqueue->head = (aqueue->head+1)%aqueue->size;
  aqueue->num--;
  return adata;
}
```



Queue Implementation Using Array

- isEmpty, isFull

```
int isEmpty(Queue *aqueue) {
  return (aqueue->num == 0);
}

int isFull(Queue *aqueue) {
  return (aqueue->num == aqueue->size);
}
```

Queue Implementation Using Array

- front, makeEmpty

```
int front(Queue *aqueue)
{
  return aqueue->data[aqueue->head];
}

void makeEmpty(Queue *aqueue)
{
  aqueue->head = 0;
  aqueue->tail = -1;
  aqueue->num = 0;
}
```

Interesting Question

- What if we don't use "num" in the queue definition?

Before

```
typedef struct {
  int *data;     //data is an array of int
  int front;
  int rear;
  int num;       //number of elements in queue
  int size;      //size of queue
}Queue;
```

After

```
typedef struct {
  int *data;     //data is an array of int
  int front;
  int rear;
  int size;      //size of queue
}Queue;
```

Interesting Question

Before

```c
//return 1 for success, 0 for fail
int createQueue(Queue *aqueue, int size)
{
  aqueue->data = (int*)malloc(sizeof(int)*size);
  if (aqueue->data == NULL)
    return 0;
  aqueue->front = 0;
  aqueue->rear = -1;
  aqueue->num = 0;
  aqueue->size = size;
  return 1;
}
```

After

```c
//return 1 for success, 0 for fail
int createQueue(Queue *aqueue, int size)
{
  aqueue->data = (int*)malloc(sizeof(int)*size);
  if (aqueue->data == NULL)
    return 0;
  aqueue->front = aqueue->rear = 0;
  aqueue->size = size;
  return 1;
}
```

Interesting Question

Before

```c
int isEmpty(Queue *aqueue) {
  return (aqueue->num == 0);
}

int isFull(Queue *aqueue) {
  return (aqueue->num == aqueue->size);
}
```

After

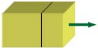
```c
int isEmpty(Queue *aqueue) {
  return (aqueue->front == aqueue->rear);
}

int isFull(Queue *aqueue) {
  return (((aqueue->rear+1)%aqueue->size) == aqueue->front);
}
```

Interesting Question

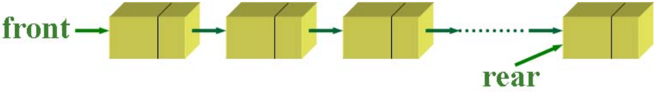
- How about enqueue() and dequeue()?
- How many data can the queue store?

Implementation Using Linked List

```c
struct node_s {
  int data;
  struct node_s *next;
};
typedef struct node_s node;
```



```c
typedef struct {
  node *front, *rear;
}Queue;
```



Implementation Using Linked List

- isEmpty

```c
int isEmpty(Queue *aqueue) {
  return (aqueue->front == NULL);
}
```

- ”isFull()” is no longer needed

Implementation Using Linked List

- enqueue

```c
void enqueue(Queue *aqueue, int adata)
{
  node *newnode = (node *)malloc(sizeof(node));
  newnode->data = adata;
  newnode->next = NULL;
  if ( aqueue->front == NULL )
    aqueue->front = aqueue->rear = newnode;
  else {
    aqueue->rear->next = newnode;
    aqueue->rear = aqueue->rear->next;
  }
}
```

Implementation Using Linked List

- dequeue

```
int dequeue(Queue *aqueue)
{
  if ( isEmpty(aqueue) ) //Queue is empty
    return -1;

  node *p = front;
  int x = aqueue->front->data;
  aqueue->front = aqueue->front->next;
  delete p;
  return x;
}
```

Implementation Using Linked List

- getFront, makeEmpty

```
int getFront(Queue *aqueue)
{
  if ( isEmpty(aqueue) ) return -1;
  return aqueue->front->data;
}

void makeEmpty(Queue *aqueue)
{
  node *p;
  while ( aqueue->front != NULL ) {
    p = aqueue->front;
    aqueue->front = aqueue->front->next;
    delete p;
  }
}
```

- When will the problem IDs be released?
  - Generally, the problem IDs will be released one day before the Online Judge is opened.
- Do we need to validate the user input?
  - No.
- Would it be alright if I use both getchar() and scanf() in the assignment?
  - Yes.
- Can we use string handling functions included in ”string.h” in our programming assignment?
  - Yes. If there are some functions that cannot be used, they will be stated explicitly.
- Are we expected to use pointer?
  - It is no a requirement. If it is needed, it will be stated explicitly.
- Do we need to store all the outputs and print them at last?
  - No. You can print the output after reading each input.

Exercise 1.13 Given a pair of integers. Calculate the summation and subtraction of these two integers.

Input The input consists of the number of test cases, m, in the first line and followed by m groups of 4 lines as inputs. The group consists of a symbol, either + or -, two lines of integers followed by a carriage return. The integer can have 100 digits and can also be negative. An example is as follows,

```
2
+
123456
111111

-
0
-10
```

Output The output should be m lines of numbers. Each line should be the summation or the difference of the two integers.

```
234567
10
```

How to read input

- scanf()
- Case 1 (Wrong)

```
scanf("%d",&noOfCase);
scanf("%c",&op);
scanf("%d",&no1);
scanf("%d",&no2);
scanf("\n");
...
```

- No "\n" in the scanf().
- If the next scanf() reads integer, no need to read the carriage return.
- If the next scanf() reads character, you need to read the carriage return.

Input file

```
2
+
123456
111111
(blank line)
-
0
-10
(blank line)
```

How to read input

- scanf()
- Case 2 (Wrong)

```
scanf("%d",&noOfCase);
scanf("%c",&cr);
scanf("%c",&op);
scanf("%d",&no1);
scanf("%d",&no2);
scanf("%c",&cr);
scanf("%c",&cr);
...
```

- You need to use char* to read the numbers.

Input file

```
2
+
123456
111111
(blank line)
-
0
-10
(blank line)
```

How to read input

- scanf()
- Case 3 (Correct)

```
scanf("%d",&noOfCase);
scanf("%c",&cr);
scanf("%c",&op);
scanf("%c",&cr);
scanf("%s",&no1);
scanf("%s",&no2);
scanf("%c",&cr);
scanf("%c",&cr);
...
```

- If the next scanf() reads char*, you can either read or not read the carriage return.

Input file

2

+

123456

111111

(blank line)

-

0

-10

(blank line)

Exercise 1.14 Given a pair of non-negative integers between 0 and 65535. Find the number of bits that are different in their respective binary representation. For example, 3 in decimal is equivalent to 0000000000000011 in binary and 1 in decimal is equivalent to 0000000000000001 in binary so that the number of bits that are different in these two binary patterns is 1.

Input The input consists of the number of test cases, m, in the first line and followed by m lines of two positive integers as inputs. For example,

```
3
1 3
100 100
65535 0
```

Output The output should be m lines of numbers.

```
1
0
16
```

Operators you may need

- ˆ (XOR)

```
no1 = 1;
no2 = 3;
no3 = no1^no2;
```

- & (AND)

```
no1 = 3;
no2 = no1&1;
no3 = no1&2;
```

Table: XOR truth table

x	y	xˆy
0	0	0
0	1	1
1	0	1
1	1	0

Table: AND truth table

x	y	x&y
0	0	0
0	1	0
1	0	0
1	1	1

Question