

东莞城市学院人工智能学院

School of Artificial Intelligence

《数据结构》

实验报告

| 学号 | 202435710252 |
| --- | --- |
| 姓名 | 钟润柯 |
| 专业、班级 | 2024级计科（专升本）2班 |
| 课程名称 | 《数据结构》 |
| 指导教师 | 李言一 |
| 学期 | 2024—2025第1学期 |
| 实验次序 | 实验三 |
| 上机时间 | 2024-11-6 周三 第5-6节 |
| 上机地点 |  |
| 分数 |  |

二○二三——二○二四学年 第一学期

实验四 队列的定义和操作

# 实验课时

课内：2课时

# 实验类型

操作性实验

# 实验目的

1.熟练掌握队列的特点（先进先出）。

2.掌握队列的定义和基本操作，如入队、出队等。

3.熟练掌握顺序（循环）队列和链队列的操作及应用。

4.能编程实现顺序（循环）队列和链队列的创建、入队和出队等。

5.逐步培养解决实际问题的编程能力。

# 实验内容和步骤

## 一、源代码

（1）队列的顺序表定义和实现的源码

定义部分、函数部分和主函数生效部分：

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  #define MAXSIZE 100  #define ElemType int  typedef struct SqQueue {      int\* base;  // 数据存储的地址      int front;  // 指向队列的第一个元素      int rear;   // 指向队列最后一个元素的下一个位置      int queuesize;  // 队列容量  } SqQueue;  void InitSqQueue(SqQueue\* Q) {      Q->base = (int\*)malloc(MAXSIZE \* sizeof(int));      if (!Q->base) {          printf("Memory allocation failed\n");          exit(-1);      }      Q->front = 0;      Q->rear = 0;      Q->queuesize = MAXSIZE;  }  int IsFullSqQueue(SqQueue\* Q) {      return (Q->rear + 1) % Q->queuesize == Q->front;  }  int IsEmptySqQueue(SqQueue\* Q) {      return Q->front == Q->rear;  } |

（2）队列的顺序表插入数据的源码：

|  |
| --- |
| int SqQueuePush(SqQueue\* Q, int e) {      if (IsFullSqQueue(Q)) {          printf("Queue is full\n");          return 0;      }      else {          Q->base[Q->rear] = e;          Q->rear = (Q->rear + 1) % Q->queuesize;          return 1;      }  }  int main() {      SqQueue Q;      printf("任务1顺序表\n");      printf("定义一个队列Q\n");      InitSqQueue(&Q);      printf("初始化队列Q InitSqQueue\n");      for (int i = 1; i <= 5; i++) {          SqQueuePush(&Q, i);      }      for (int i = 1; i <= 5; i++) {          int n;          n = SqQueuePop(&Q);      }      printf("入队第1次：3 2 1\n");      SqQueuePush(&Q, 3);      SqQueuePush(&Q, 2);      SqQueuePush(&Q, 1);      int n;      n = SqQueuePop(&Q);      printf("出队1次：%d\n", n);      n = SqQueuePop(&Q);      printf("出队2次：%d\n", n);      printf("入队第2次：4 5 6 7\n");      SqQueuePush(&Q, 4);      SqQueuePush(&Q, 5);      SqQueuePush(&Q, 6);      SqQueuePush(&Q, 7);      for (int i = 1; i <= MAXSIZE + 1; i++) // 注意这里会越界          printf("Queue's 出队%d\n", SqQueuePop(&Q));  } |

（3）队列的顺序表删除数据的源码：

|  |
| --- |
| int SqQueuePop(SqQueue\* Q) {      if (IsEmptySqQueue(Q)) {          printf("Queue is empty\n");          exit(-1);      }      else {          int e = Q->base[Q->front]; // 取出对头元素          Q->front = (Q->front + 1) % Q->queuesize;          return e;      }  } |

（4）队列的链式表示定义和实现的源码

定义部分、函数部分和主函数生效部分：

|  |
| --- |
| // 链队列  #include <stdio.h>  #include <stdlib.h>  #define Elemtype int  typedef struct QNode{      Elemtype data;      struct QNode\* next;  } QNode;  typedef struct {      QNode\* front;      QNode\* rear;  } LinkQueue;  // 初始化链队列  void InitQueue(LinkQueue\* Q) {      Q->front = Q->rear = (QNode\*)malloc(sizeof(QNode));      if (!Q->front) {          printf("Memory allocation failed\n");          exit(-1);      }      Q->front->next = NULL;  }  // 判断队列是否为空  int IsEmpty(LinkQueue\* Q) {      return Q->front == Q->rear;  } |

（5）队列的链式表示插入数据的源码

|  |
| --- |
| // 入队操作  void QueuePush(LinkQueue\*Q, Elemtype e) {      QNode\* newNode = (QNode\*) malloc (sizeof(QNode));      if (!newNode) {          printf("Memory allocation failed\n");          exit(-1);      }      newNode->data = e;      newNode->next = NULL;      Q->rear->next = newNode;      Q->rear = newNode;  } |

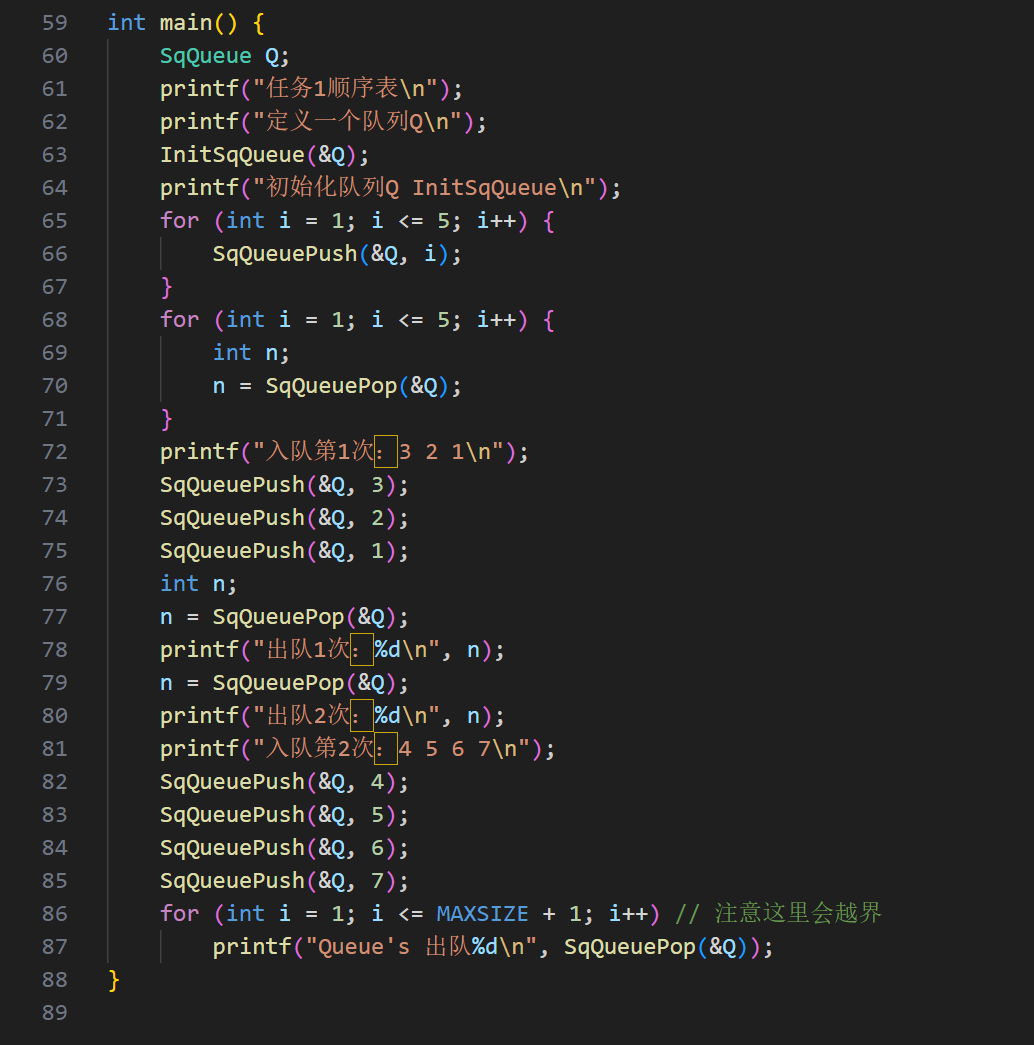
（6）队列的链式表示删除数据的源码

|  |
| --- |
| // 出队操作  int QueuePop(LinkQueue\* Q) {      if (IsEmpty(Q)) {          printf("Queue is empty\n");      }      QNode\* p = Q->front->next;      Elemtype e = p->data;      Q->front->next = p->next;      if (Q->rear == p) {          Q->rear = Q->front;      }      free(p);      return e;  } |

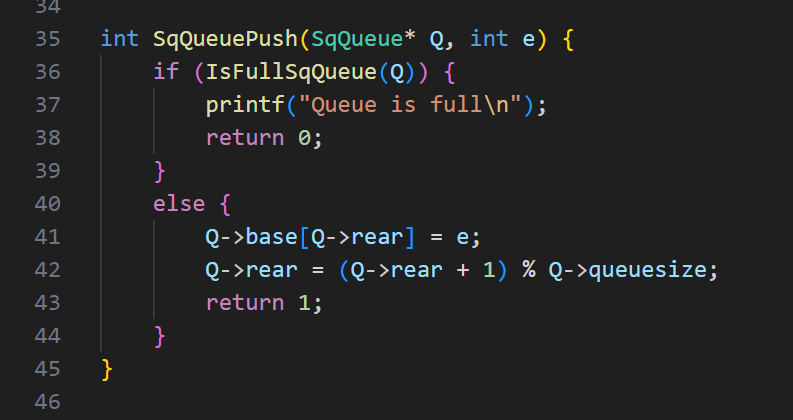
## 二、程序运行结果截图

（1）队列的顺序表定义和实现的截图：

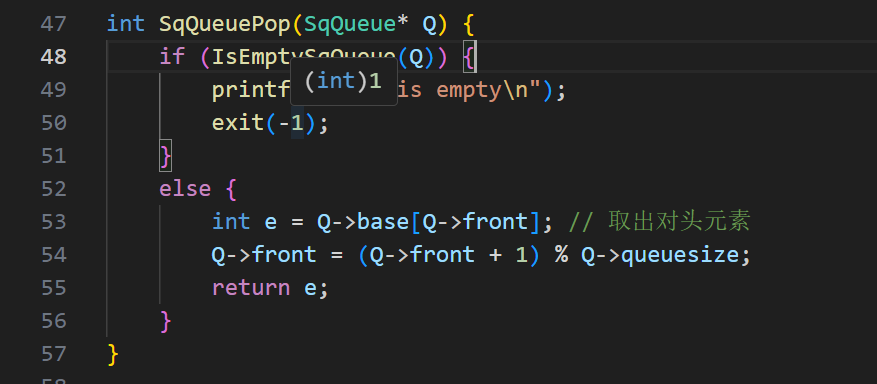




（2）队列的顺序表插入数据的截图：

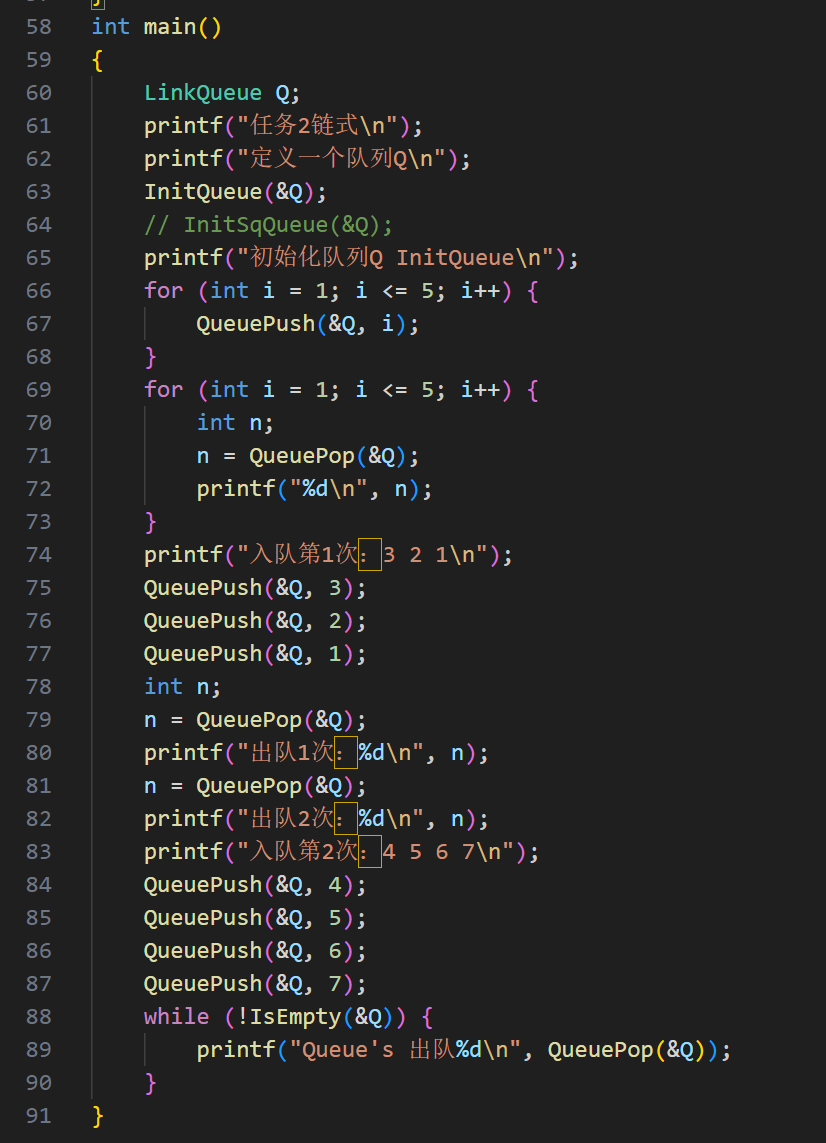


（3）队列的顺序表删除数据的截图：

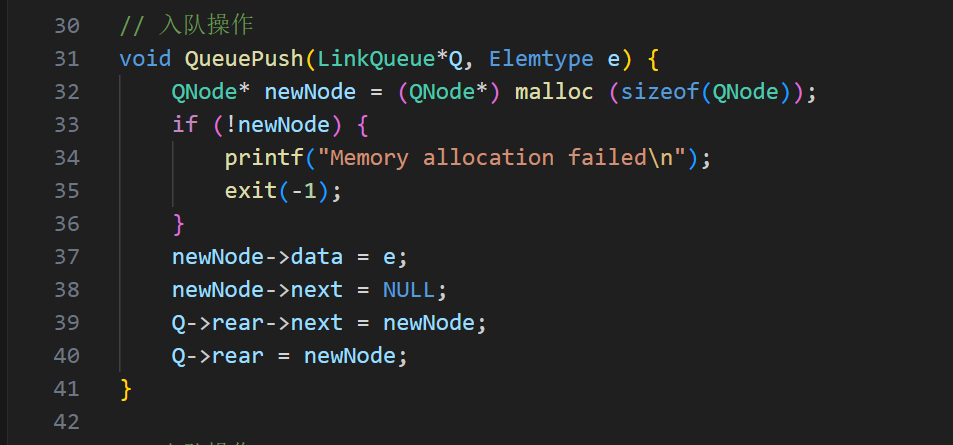


（4）队列的顺序表定义和实现的截图：

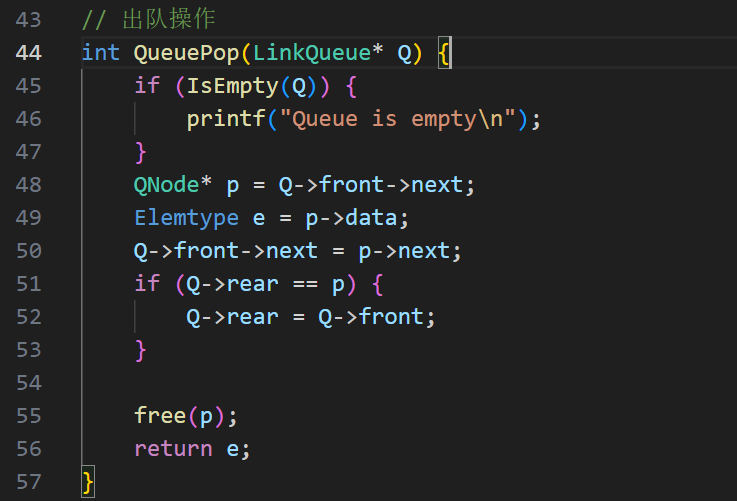




（5）队列的顺序表插入数据的截图：



（6）队列的顺序表删除数据的截图：



## 三、实验总结

通过实验，深入理解了队列的特性和实现方式，掌握了循环队列和链队列的基本操作及其应用场景。实验过程中，注意了队列空满状态的判断和处理，确保了程序的健壮性。