# 任务3加入相关算法

## 1.1 任务要点

使用程序设计要求的算法

## 1.2 任务内容

基数排序，二分查找

## 1.3 任务实现

**流程图：**

基数排序->二分查找

**实现效果：**

使用基数排序和二分查找可以快速的对书籍编号进行查询

**核心代码：**

void radixSort(Book book[],int n)

{

int max = findMax(book,0,n-1);

int i,j,k,num,index;

for(i=1;max/i>0;i=i\*10)

{

Book buckets[n][10];

for(j=0;j<n;j++)

{

for(k=0;k<10;k++)

{

buckets[j][k].num = -1;

}

}

for(j=0;j<n;j++)

{

num = (book[j].num/i)%10;

buckets[j][num].num = book[j].num;

strcpy(buckets[j][num].name,book[j].name);

strcpy(buckets[j][num].author,book[j].author);

strcpy(buckets[j][num].publish,book[j].publish);

strcpy(buckets[j][num].date,book[j].date);

buckets[j][num].price = book[j].price;

}

index = 0;

for(j=0;j<10;j++)

{

for(k=0;k<n;k++)

{

if(buckets[k][j].num!=-1)

{

book[index].num = buckets[k][j].num;

strcpy(book[index].name,buckets[k][j].name);

strcpy(book[index].author,buckets[k][j].author);

strcpy(book[index].publish,buckets[k][j].publish);

strcpy(book[index].date,buckets[k][j].date);

book[index].price = buckets[k][j].price;

index++;

}

}

}

}

}

int binary\_search(Book book[],int len,int book\_num)

{

int low = 0;

int hight = len - 1;

int midder;

while(low <= hight)

{

midder = (low + hight) / 2;

if(book[midder].num == book\_num)

{

return midder;

}

else if(book[midder].num > book\_num)

{

hight = midder - 1;

}

else if(book[midder].num < book\_num)

{

low = midder + 1;

}

}

return -1;

}