数据结构

▼ 链表

```
#include "iostream"
using namespace std;
typedef struct Lnode
{
    char data;
    struct Lnode *next;
}Lnode,*link;
void init(link &l)
{
    l=(link)malloc(sizeof(Lnode));
    l->next=NULL;
}
void createwei(link &l,int n)//尾插
{
    FILE *fp=fopen("C:/Users/Administrator/Desktop/11.txt"
    init(l);
    link pre=l;
    for(int i=0; i<n; i++)
    {
        link p=(link)malloc(sizeof(Lnode));
        fscanf(fp,"%c ",&(p->data));
        pre->next=p;
        pre=p;
    }
    pre->next=NULL;
    fclose(fp);
}
void creattou(link &l,int n)//头插
{
    FILE *fp=fopen("C:/Users/Administrator/Desktop/11.txt"
    init(l);
    l->next=NULL;
```

```
    for(int i=0; i<n; i++)
    {
        link p=(link)malloc(sizeof(Lnode));
        fscanf(fp,"%c ",&p->data);
        p->next=l->next;
        l->next=p;
    }
}
void insert(link &l,int pos,char c)//插入
{
    int j=0;
    link p=l;
    while(p&&j<pos-1)
    {
        j++;
        p=p->next;
    }
    if(!p||j>pos-1)
    {
        return;
    }
    link s=(link)malloc(sizeof(Lnode));
    s->data=c;
    s->next=p->next;
    p->next=s;
}
int locate(link l,char c)//查询
{
    int j=1;
    link p=l->next;
    while(p&&p->data!=c)
    {
        j++;
        p=p->next;
    }
    if(!p)
    {
        return -1;
```

```
    }
    return j;
}

int main()
{

}
```

▼ 循环队列

```
#include<iostream>
#define intsize 100
typedef struct tagqueue//建立列表
{
    int* ele;
    int front,id;
}queue;
void initqueue(queue &q)//初始化
{
    q.ele=(int*)malloc(intsize*sizeof(int));
    q.front=q.id=0;
}
void enqueue(queue &q,int a)//入队列
{
    q.ele[q.id]=a;
    q.id=(q.id+1)%100;
}
void dequeue(queue &q,int &a)//出队列
{
    a=q.ele[q.front];
    q.front=(q.front+1)%intsize;
}
bool qempty(queue q)//判空
{
    return q.front==q.id;
}
bool qf(queue q)//判满
```

```
{
    return q.front==(q.id+1)%intsize;
}
int main()
{

}
```

▼ 串

▼ BF

```
#include<iostream>
#define intsize 100
typedef struct tagstring
{
    char *c;
    int len;
}string;
int idxBF(string t,string p,int pos)
{
    int i=pos,j=0;
    while(i<=t.len-p.len&&j<p.len)
    {
        if(p[j]==t[i+j])
        {
            j++;
        }
        else
        {
            i++;
            j=0;
        }
    }
    if(j==p.len)
    {
        return i;
    }
    return -1;
```

```
}
int main()
{

}
```

▼ KMP

```

```

▼ 找不着

▼ 顺序查找

```
#include "iostream"
using namespace std;
typedef long long ll;
typedef struct tagInt
{
    int key;
    int info;
}Int;
typedef struct
{
    Int elem[1000];
    int len;
}sstable;
int search(sstable s,int key)
{
    for(int i=s.len;i>0;i--)
    {
        if(key==s.elem[i].key)
        {
            return i;
        }
    }
    return -1;
}
int main()
{
```

```
}
```

平均查找长度：$ASL=(n+1)/2$

$O(n)$

▼ 关键字有序

▼ 有序表查找

▼ 二分

```
#include "iostream"
using namespace std;
typedef long long ll;
typedef struct tagInt
{
    int key;
    int info;
}Int;
typedef struct
{
    Int elem[1000];
    int len;
}sstable;
int search(sstable s,int key)
{
    for(int i=s.len;i>0;i--)
    {
        if(key==s.elem[i].key)
        {
            return i;
        }
    }
    return -1;
}
int BS(sstable s,int key)
{
```

```
    int l=1,r=s.len;
    int mid=(l+r)/2;
    while(l<=r)
    {
        if(key==s.elem[mid].key)
        {
            return mid;
        }
        else if(key<s.elem[mid].key)
        {
            r=mid-1;
        }
        else
        {
            l=mid+1
        }
    }
}
int main()
{

}
```

- ▼ 梭哈查找
- ▼ AVL

```

```

- ▼ *树

- ▼ 排序
  - ▼ 插排$O(n^2)$

```
#include<iostream>
using namespace std;
typedef struct tagrecord
{
    int key;
```

```
}rcdtype;
void insert(rcdtype r[],rcdtype x,int i)
{
    int j;
    if(x.key<r[i].key)
    {
        for(j=i;j>=0;j--)
        {
            if(x.key<r[j].key)
            {
                r[j+1]=r[j];
            }
        }
        r[j+1]=x;
    }
}
void insertsort(rcdtype r[],int n)
{
    rcdtype temp;
    for(int i=1;i<n;i++)
    {
        temp=r[i];
        insert(r,temp,i-1);
    }
}
int main()
{

}
```

▼ 选排$O(n^2)$

```
#include<iostream>
using namespace std;
typedef struct tagrecord
{
    int key;
}rcdtype;
```

```
int xiaoid(rcdtype r[],int n,int i)
{
    int id=i,Min=r[i].key;
    for(int j=i+1;j<n;j++)
    {
        if(r[j].key<r[id].key)
        {
            id=j;
        }
    }
    return id;
}
void selectsort(rcdtype r[],int n)
{
    int id;
    for(int i=0;i<=n-2;i++)
    {
        id=xiaoid(r,n,i);
        swap(r[i],r[id]);
    }
}
int main()
{

}
```

▼ bubble

```

```

▼ 希儿

```

```

▼ 快排

```
#include<iostream>
using namespace std;
typedef struct tagrecord
{
```

```
    int key;
}rcdtype;
int partition(rcdtype e[],int l,int r)
{
    rcdtype p=e[l];
    while(1)
    {
        while(l<r&&e[r].key>=p.key)
        {
            r--;
        }
        if(l==r)
        {
            break;
        }
        if(l==r)
        {
            break;
        }
        e[l++]=e[r];
        while(l<r&&e[l].key<=p.key)
        {
            l++;
        }
        if(l==r)
        {
            break;
        }
        e[r--]=e[l];
    }
    e[l]=p;
    return l;
}
void Sort(rcdtype r[],int s,int n)
{
    if(s<n)
    {
```

▼ 归并

▼ 堆排　建堆$O(n)$

```
#include<iostream>
using namespace std;
typedef struct tagrecord
{
    int key;
}rcdtype;
void adjust(rcdtype r[],int s,int t)
{
    rcdtype temp=r[s];
    int f=s;
    int c=2*f;
    while(c<=t)
    {
        if(c<t&&r[c+1].key>r[c].key)
        {
            c++;
        }
        if(temp.key>r[c].key)
        {
            break;
        }
        r[f]=r[c];
        f=c;
        c=2*c;
        r[f]=temp;
    }
}
void buildheap(rcdtype r[],int n)
{
    for(int j=n/2;j>=1;j--)
    {
        adjust(r,j,n);
    }
```

```
}
void heapsort(rcdtype r[],int n)
{
    for(int i=n;i>=2;i--)
    {
        swap(r[n],r[1]);
        adjust(r,1,n-1);
    }
}
int main()
{

}
```

```
#include "iostream"
using namespace std;
typedef struct taghnode
{
    int key;
}hnode;
typedef struct tagheap
{
    hnode *node;
    int length;
}heap;
void adjust(hnode elem[],int s,int t)
{
    int c=2*s;
    hnode temp=elem[s];
    while(c<=t)
    {
        if(c<t&&elem[c+1].key>elem[c].key)
        {
            c++;
        }
        if(temp.key>=elem[c].key)
        {
            break;
```

```
        }
        elem[s]=elem[c];
        s=c;
        c=2*c;
    }
    elem[s]=temp;
}
int main()
{

}
```