# c++期末算法大汇总

## 剥离算法

### 最简单剥离1.0版

```
int seperation(int num){
    return n%10

}
```

### 剥离后逆序输出

```
int seperation(int num){
    int n=0;
    while (num>0){
        n=n*10+num%10;
        num/=10;

    }
    return n;

}
```

## 判断质数

```
//是质数输出为1，否则为0
int isPrime(int num){
    int flag=1;
    //下面这个循环实际可以优化范围到根号n
    for (int i=2;i<n;i++){
        if (num%i==0){
            flag=0;
            break;
        }

    }
    return flag;
}
```

## 求最大公因数

### 辗转相除法（递归版）

```
int gcd(int a,int b){
    if (a%b==0){
        return b;
    }
    else
        return gcd(b,a%b);
}
```

### 辗转相除法（普通版）

```
int gcd(int a,int b){
    while (b!=0){
        int c=a%b;
        a=b;
        b=c;
    }
    return a;
}
```

### 暴力枚举法

```
int gcd(int a,int b){
    int flag=0;
    if (a>b){
        int tmp;
        tmp=a;
        a=b;
        b=tmp;
    }
    for (int i=2;i<=a;i++){
        if (b%i==0){
            flag=1;
            return b;
        }
    }
    if (flag==0)
        return 1;
}
```

## 字符串处理函数

### 统计字符个数

```
int myStrlen (char *s){
    int i=0;
    while (*(s+i)!='\0'){
        i++;
    }
    return i-1;
}
```

## 字符串的比较

```
int myStrcmp(char *s1,char *s2){
    int flag=0;
    int i=0;
    while (*(s1+i)!='\0'&&*(s2+i)!='\0'){
        if (*(s1+i)>*(s2+i)){
            flag=1;
            break;
        }
        else if(*(s2+i)>*(s1+i)){
            flag=-1;
            break;
        }
        else
            i++;
    }
    return flag;
}
```

## 字符检索

```
char *myIndex(char *p,char c){
    char *point;
    int flag=0;
    point=p;
    while (*point!='\0'){
        if (*point==c){
            flag=1;
            break;
        }
        point++;
    }
    if (flag==0)
        return NULL;
    else
        return *point;
}
```

## 字符串连接

```
char *myStrcat(char s1[],char s2[]){
    int i=0;
    char s3[10000];
    while (s1[i]!='\0'){
        s3[i]=s1[i];
        i++;
    }
    int j=0;
    while (s2[j]!='\0'){
        s3[i]=s2[j];
        i++;
        j++;
```

```
    }
    s3[i]='\0';
    return s1;
}
```

### 字符串复制

```
char *strcpy(char s1[],char s2[]){
    int i=0;
    while (s2[i]!='\0'){
        s1[i]=s2[i];
        i++;
    }
    s1[i]='\0';
    return s1;
}
```

### 字符串大小写转换

```
void toggle(char *s){
    int i=0;
    while (*(s+i)!='\0'){
        if (*(s+i)>='a'&&*(s+i)<='z')
            *(s+i)=*(s+i)-'a'+'A';
        else if(*(s+i)>='A'&&*(s+i)<='Z')
            *(s+i)=*(s+i)-'A'+'a';
        i++;
    }
}
```

## 十进制转二进制

### 数组解法

```
void convert(int num){
    int s[100];
    int i=0;
    while (num>0){
        s[i]=num%2;
        num/=2;
        i++;
    }
    for (i;i>=0;i--){
        cout<<s[i-1];
    }
    cout<<endl;
}
```

### 递归解法

```
void convert(int num){
    if (num>0){
        convert(num/2);
        cout<<num%2;
    }
}
```

## 判断回文数

```
int hw(int num){
    int digit[100];
    int n=num;
    int i=0;
    while (num>0){
        digit[i]=num%10;
        num/=10;
        i++;
    }
    int reverseNum=0;
    for (int j=0;j<i;j++){
        reverseNum=reverseNum*10+digit[i];
    }
    if (reverNum==n)
        return 1;
    else
        return -1;
}
```

## 一维数组处理

### 数组求平均数

```
double avg(int s[],int n){
    double avg=0;
    for (int i=0;i<n;i++){
        avg+=s[i]*1.0/n;
    }
    return avg;
}
```

### 数组中找最大数

```
double max(int s[],int n){
    double max=s[0];
    for (int i=0;i<n;i++){
        if (s[i]>max){
            max=s[i];
        }
    }
}
```

# 查找算法

## 顺序查找

```
int search(int s[],int n,int find){
    int flag=0;
    int i=0;
    for (i=0;i<n;i++){
        if (s[i]==find){
            flag=1;
            break;
        }
    }
    if (flag==0)
        return -1;
    else
        return i;
}
```

## 二分查找(普通版)

```
//假设数组元素为升序
int binarySearch(int s[],int n,int find){
    int left=0,right=n-1;
    int mid=(left+right)/2;
    int flag=0;
    while (left<=right){
        if (s[mid]==find){
            flag=1;
            break;
        }
        else if(s[mid]<find)
            mid=left+1;
        else
            mid=right-1;
    }
    if (flag==0)
        return -1;
    else
        return mid;
}
```

## 二分查找（递归版）

```
//假设数组元素为升序
int binarySearch(int s[],int left,int right,int find){
    if (left<=right){
        int mid=(left+right)/2;
        if (s[mid]==find)
            return mid;
        else if(s[mid]<find)
            return binarySearch(int s[],int mid+1,int right,int find);
        else
            return binarySearch(int s[],int left,int mid-1,int find);
    }
    return -1;
}
```

# 排序算法

## 冒泡排序

```
//升序
void bubbleSort(int s[],int n){
    for (int i=0;i<n-1;i++){
        for (int j=0;j<n-i-1;j++){
            if (s[j]>s[j+1]){
                int tmp=s[j];
                s[j]=s[j+1];
                s[j+1]=tmp;
            }
        }
    }
}
```

## 选择排序

```
//升序
void selectionSort(int a[],int n){
    int min;
    for (int i=0;i<n;i++){
        min=i;
        for (int j=i+1;j<n;j++){
            if (a[j]<a[min])
                min=j;
        }
        int tmp;
        tmp=a[j];
        a[j]=a[min];
        a[min]=tmp;
    }
}
```