约瑟夫问题

递归法

```
int Joseph(int m, int n)    //m 为总人数，n 为报数间隔
{
    if (m == 1)    return 0;
    else    return ((Joseph(m - 1, n) + n) % m);
}
```

```
#include<iostream>
#include<ctime>
using namespace std;
int main()
{
    int position(int alive);
    clock_t start, finish;
    start = clock();
    int alive, pos, count;
    for (count = 0; count < 100000; count++)  //本循环便于用放大法更精确计算程序运行的耗时
    {
        alive = 40;
        pos = position(alive);
    }
    finish = clock();
    cout << "这个位置的编号为" << pos << endl;
    cout << "程序共用时" << (finish - start) / 100000.0 << "毫秒" << endl;
    return 0;
}
int position(int alive)
{
    if (alive > 1) return (position(alive - 1) + 3) % (alive);  //算法解析见实验报告
    if (alive == 1) return 0;
}
```

指针递推法

```
#include<iostream>
using namespace std;
int main()
{
    void del(int *p, int n);
    int num[50], i, n;   //用数组 num 记录人的状态，1 表示在圈内，0 表示已退出；n 表示参与的总人数
    cout << "总人数为：";
    cin >> n;
    if (n >= 50) cout << "请输入小于 50 的正整数！" << endl;
    else
    {
        for (i = 0; i < n; i++)
            num[i] = 1;   //初始化这 n 个人的状态
        del(num, n);   //执行退出与打印最后留下的人的编号的函数
    }
    return 0;
}
void del(int *p, int n)
{
    int remain, code, pos;   //remain 表示场上留下的人数，code 表示当前所报的数，pos 表示报数者的位置(即编号)
    remain = n; code = 0; pos = -1;  //初始化数据
    while (remain > 1)  //报数进行的条件
    {
        pos++;
        if (pos == n) pos = 0;  //确保编号最末的人与最前的人报数的衔接
        if (*(p + pos) == 1) code++;
        if (code == 3)   //处理报 3 的人退出
        {
            *(p + pos) = 0;   //将编号为 pos 的人状态置 0
            remain--;   //场上人数减 1
            code = 0;   //报数重新置 0
        }
    }
    if (remain == 1)  //确定最后留下的人的编号
    {
        if (pos == -1) pos++;   //考虑到 n=1 的情况，上面 while 循环体不执行，pos++应另外执行
        while (!*(p + pos))   //寻找还在场上的人
        {
            pos++;
            if (pos == n) pos = 0;
        }
    }
    cout << "最后留下来的人的初始编号为：" << pos << endl;
}
```

高斯消元法

```
#include<iostream>
#include<iomanip>
using namespace std;
double a[4][5] = { { 1.1161,0.1254,0.1397,0.1490,1.5471 },  { 0.1582,1.1675,0.1768,0.1871,1.6471 },
{ 0.2368,0.2471,0.2568,1.2671,1.8471 },  { 0.1968,0.2071,1.2168,0.2271,1.7471 } };
double x[4];
int main()
{
    void printMatA();   //输出系数矩阵的函数
    void printMatB();   //输出常数矩阵的函数
    void solve();    //高斯消去函数
    void printX();   //输出方程组的解的函数
    cout << setiosflags(ios::fixed) << setprecision(4);  //统一输出格式
    cout << "-------------Original Mats--------------" << endl;
    printMatA();
    printMatB();
    solve();
    cout << endl << "------Mats After Gauss Elimination------" << endl;
    printMatA();
    printMatB();
    cout << endl << "---Solution---" << endl << endl;
    printX();
    return 0;
}


void printMatA()
{
    cout << "MAT A =" << endl;
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
            cout << setw(10) << a[i][j];
        cout << endl;
    }
    cout << endl;
}
void printMatB()
{
    cout << "MAT B =" << endl;
    for (int i = 0; i < 4; i++)
        cout << setw(10) << a[i][4];
    cout << endl << endl;
}
void solve()
{
    int i, j, k, t;
    double temp;
//下面求上三角矩阵
    for (k = 0; k < 4; k++)
    {
     //列选主元
        t = k;
        for (i = k + 1; i < 4; i++)
            if (a[i][k] > a[t][k]) t = i;
        if (t != k)
        {
            for (j = 0; j < 5; j++)
            {
                temp = a[k][j];
                a[k][j] = a[t][j];
                a[t][j] = temp;
            }
        }
        //归一化
            temp = a[k][k];
            for (j = k; j < 5; j++)
                a[k][j] = a[k][j] / temp;
         //消去
            for (i = k + 1; i < 4; i++)
            {
                temp = a[i][k];
                for (j = k; j < 5; j++)
                    a[i][j] = a[i][j] - temp * a[k][j];
            }
    }
//下面进行回代，解出 x
    for (i = 3; i >= 0; i--)
    {
        x[i] = a[i][4];
        for (j = i + 1; j < 4; j++)
            x[i] -= x[j] * a[i][j];
    }
}
void printX()
{
    for (int i = 0; i < 4; i++)
        cout << "x(" << i + 1 << ") = " << x[i] <<
endl;
}
```

IP 转换

在网络编程时，经常需要把IP地址转换为计算机内部的整型数来处理。C++系统提供的`atoi()`就是实现该功能。参考该函数，编写另一个函数（如`aton()`），其功能是将输入的IPv4地址（字符串，例如166.111.64.89 ）字符串转换为无符号整数输出。注：`aton()`函数应返回无符号十进制整型数，然后在`main`函数中将该返回值转换为32位二进制数输出。

要求：
- 输入参数：`str`，输入字符串
- 返回值：转换结果，若`str`无法转换成整数，返回0
- 函数申明：`unsigned int aton(const char str[]);`

例如：输入`"166.111.64.89"`，`aton`函数返回值应为2792308825 ($166*2^{24} + 111*2^{16} + 64*2^8 + 89 = 2792308825$)，最后在`main`中转为`10100110011011110100000001011001`输出。

```
#include<iostream>
#include<string>
#include<sstream>
#include<cmath>
using namespace std;
string a[4];
bool flag = true;
int main()
{
    string str, B;
    unsigned int D;
    void check_store(string);
    unsigned int aton(string);
    string switchDB(unsigned int);
    cin >> str;
    check_store(str);
    D = aton(str);
    if (!flag)
        cout << "0 (input error)";
    else
    {
        B = switchDB(D);
        cout << "Decimal: " << D << endl;
        cout << "Binary: " << B << endl;
    }
    return 0;
}
void check_store(string str)  //用于判断输入格式的正确性并将每段数字存入字符串 a 中
{
    int i, j = 0;
    bool checknum(char);
    for (i = 0; i < 4; i++)
    {
        if (!checknum(str[j]))
        {
            flag = false;
            break;
        }
        else
        {
            while (checknum(str[j]))
            {
                a[i] = a[i] + str[j];
                j++;
            }
            if ((i < 3) && (str[j] != '.'))
            {
                flag = false;
                break;
            }
```

```
            j++;
        }
    }
    if (j < str.length() + 1) flag = false;
}
bool checknum(char c)  // 用于判断字符 c 是否是数字
{
    if (c >= 48 && c <= 57) return true;
    else return false;
}
unsigned int aton(string str)  //用于将输入的 IP 地址(字符串 str)转换为十进制数 D
{
    double D = 0;
    stringstream ss;  //使用字符流将字符串转换为整型数值
    int b[4];
    for (int i = 0; i < 4; i++)
    {
        ss << a[i];
        ss >> b[i];
        if (b[i] > 255)  //判断每段数字的范围是否符合 IP 地址规范
        {
            flag = false;
            break;
        }
        D += b[i] * pow(2, 8 * (3 - i));
        ss.clear();  //重复使用 ss 以提高效率
    }
    return D;
}
string switchDB(unsigned int D)  //用于将十进制数 D 转换为 32 位的二进制数 B
{
    string B;
    int r, i = 0, len;
    while (D)
    {
        r = D % 2;
        B = (char)(r + 48) + B;
        D /= 2;
        i++;
    }
    len = B.length();
    if (len < 32)  //补齐 32 位二进制数首位的 0
        for (int i = 0; i < (32 - len); i++)
            B = "0" + B;
    return B;
}
```

元音翻转

对于一个由a-z（小写）组成的字符串，将其中的元音反转，而辅音不反转。如对于字符串`"hello"`，替换后的字符为`"holle"`。（注：元音为a, e, i, o, u）

要求：输入一个由a-z（小写）组成的字符串，输出反转后的字符串。

```
input: hello
output: holle
```

提示：本题通过字符串数组和循环分支能够解决。参考算法：首先遍历字符串，将其中所有的元音存入字符串数组中，之后再次遍历字串，将其中所有的元音反序替换。

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string word;
    char temp;
    int len, i, j = 0, k = 0;
    int pos[1000];
    bool check(char c);
    cout << "input: ";
    cin >> word;
    len = word.length();
    for (i = 0; i < len; i++)
    {
        if (check(word[i]))
        {
            k += 1;
            pos[j] = i;
            j++;
        }
    }
    for (i = 0; i < k/2; i++)
    {
        temp = word[pos[i]];
        word[pos[i]] = word[pos[k - i - 1]];
        word[pos[k - i - 1]] = temp;
    }
    cout << "the vowel-reversed form: " << word << endl;
    return 0;
}
bool check(char c)
{
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') return 1;
    else return 0;
}
```

字母排序

输入一段由英文字母(区分大小写)组成的字符串，将其按ASCII码从大到小顺序输出。

要求：输入一个由英文字母（区分大小写）组成的字符串，输出符合要求的字符串。

```
input: aggregate
output: trgggeeaa
```

参考算法：本题通过字符串数组、循环分支能够求解。算法可以为冒泡排序等排序算法。若同学不会冒泡排序，则根据出现的字符仅在英文字母范围，遍历法寻找字符串中从z到A的字母，也可以求得。

```
#include<iostream>
#include<string>
using namespace std;
string s;
int main()
{
    int asc[10000], len, i;
    void sort(int array[], int len);
    cout << "input: ";
    cin >> s;
    len = s.length();
    for (i = 0; i < len; i++)
        asc[i] = s[i];
    sort(asc, len);
    cout << "output: " << s << endl;
    return 0;
}
void sort(int array[], int len)
  //用选择排序法对相关联的数组 array[len]与字符串 s 进行降序排序
{
    int i, j, k, int_t;
    char str_t;
    for (i = 0; i < len - 1; i++)
    {
        k = i;
        for (j = i + 1; j < len; j++)
            if (array[j] > array[k]) k = j;
        if (k != i)
        {
            int_t = array[i];
            array[i] = array[k];
            array[k] = int_t;
            str_t = s[i];
            s[i] = s[k];
            s[k] = str_t;
        }
    }
}
```

异位字符串

从键盘输入两个字符串，判断它们是否属于异位字符串。所谓异位字符串，就是把一个字符串中字母的顺序改变，得到的字符串。例如：`s = "anagram"`，`t = "nagaram"`，是异位字符串。`s = "rat"`，`t = "car"`，不是异位字符串。为了简化，假设字符串只包含小写英文字母。

要求：如果是异位字符串输出1，否则输出0。

```
#include<iostream>
#include<string>
using namespace std;
//本题借用了上一题的方法，先将字符串各字符按 ASCII 码降序排序，再进行比较
string s;
int main()
{
    string s1, s2;
    string scan(void);
    cout << "input s1: ";
    s1 = scan();
    cout << "input s2: ";
    s2 = scan();
    if (s1 == s2) cout << "1" << endl;
    else cout << "0" << endl;
    return 0;
}
string scan(void)
{
    int asc[10000], len, i;
    void sort(int array[], int len);
    cin >> s;
    len = s.length();
    for (i = 0; i < len; i++)
        asc[i] = s[i];
    sort(asc, len);
    return s;
}
void sort(int array[], int len)
  //用选择排序法对相关联的数组 array[len]与字符串 s 进行降序排序
{
    int i, j, k, int_t;
    char str_t;
    for (i = 0; i < len - 1; i++)
    {
        k = i;
        for (j = i + 1; j < len; j++)
            if (array[j] > array[k]) k = j;
        if (k != i)
        {
            int_t = array[i];
            array[i] = array[k];
            array[k] = int_t;
            str_t = s[i];
            s[i] = s[k];
            s[k] = str_t;
        }
    }
}
```

字符串排序（动态分配）

定义一个指向字符串的指针数组，用一个函数完成N个不等长字符串的输入，使得**指针数组元素依次指向每一个输入的字符串**。设计一个完成N个字符串按升序的排序函数（在排序过程中，要求**只交换指向字符串的指针，不交换字符串**）。在主函数中实现对排序后的字符串的输出。假设已知字符串的最大为80字节，根据实际输入的字符串长度来分配存储空间。

```
#include<iostream>
#include<string>
#include<malloc.h>
using namespace std;
int main()
{
    void input(char **);
    void sort(char **, int);
    int N, i;
    char **c;
    cout << "输入字符串的数目为：";
    cin >> N;
    c = (char **)malloc(sizeof(char *)*N);
    for (i = 0; i < N; i++)
    {
        cout << "String " << i + 1 << ": ";
        input(c + i);
    }
    sort(c, N);
    cout << endl << "排序后的字符串：" << endl;
    for (i = 0; i < N; i++)
        cout << *(c + i) << endl;
    for (i = 0; i < N; i++)
        free(*(c + i));   //释放内存
    return 0;
}
void input(char **str)  //输入字符串数组函数
{
    char inputstr[80];
    cin >> inputstr;
    int len = strlen(inputstr);
    *str = (char*)malloc(sizeof(char)*(len+1));
    strcpy_s(*str, len + 1, inputstr);
}
void sort(char **str, int n)  //字符串升序选择排序函数
{
    int i, j, k;
    char *temp;
    for (i = 0; i < n; i++)
    {
        k = i;
        for (j = i + 1; j < n; j++)
            if (strcmp(*(str + k), *(str + j)) > 0)
                k = j;
        if (k != i)
        {
            temp = *(str + k);
            *(str + k) = *(str + i);
            *(str + i) = temp;
        }
    }
}
```

Emoji

通常一个基于转义符的emoji表情输入由以下三部分构成：
转义符 + 表情名称 + 终止符
以新浪微博为例，当微博正文读取到一个转义符“[”时，它与终止符“]”之间的文字将作为表情名称在表情库中进行搜索，如果存在匹配表情，则输出显示。
注意，如果在一段语句中存在多个转义符和一个终止符，那么以离终止符最近的一个转义符作为表情的起始标志。

- 要求：编写程序，在输入的一句文字中，输出被转义符括起来的表情名称文字。
- 输入：首先输入转义符，然后输入终止符，二者均为临时指定的任意半角标点符号，随后输入一行任意的文字，由英文、数字和符号组成。文字最长不超过140个字符。
- 输出：被转义符括起来的表情名称文字，如有多个表情名称，则分行输出。

```
#include<iostream>
using namespace std;
int main()
{
    char start, end, text[140];
    int len, i = 0;
    char *p, *pos=NULL;
    bool flag = false;
    cout << "start: ";
    cin >> start;   //输入转义符
    cout << "end: ";
    cin >> end;   //输入终止符
    cout << "text: ";
    cin.ignore();
    gets_s(text);   //输入可能含有空格的文本
    len = strlen(text);
    p = text;
    cout << endl << "【Emoji】" << endl;
    while (i < len)
    {
        if (*(p + i) == start)  //记录遇到终止符前的最后一个转义符的位置
        {
            i++;
            pos = p + i;
            flag = true;
        }
        else if (flag && (*(p + i) == end))  //遇到终止符后输出 emoji
        {
            for (; pos < (p + i); pos++)
                cout << *(pos);
            cout << endl;
            flag = false;
            i++;
        }
        else i++;
    }
    return 0;
}
```

学生成绩系统（结构体）

现有某班有10名学生，每个学生的信息包括：学号（ID）、姓名（name）、性别（sex）、出生日期（birthday）和成绩（score，浮点数）。编写程序，实现如下功能：

1. 在主函数`main()`中设计一个结构体数组，将从键盘输入10名学生信息保存在该数组中；
2. 编写一个求平均值的函数，能对10名学生的成绩求平均值；
3. 编写一个排序和显示函数，能对10名学生按照成绩的高低排序，并在屏幕显示排序结果；

```
#include<iostream>
#include<iomanip>
using namespace std;
struct date   //定义结构体以表示“日期”
{  int year;  int month;  int day; };
struct student   //定义结构体以表示“学生信息”
{  long long int id;  char name[30];  char sex;  date birthday;  float score; };
int main()
{
    const int n = 10;
    float avr(struct student *, int);
    void sort_output(struct student *, int);
    student stu[n];
    int i;  float avr_score;
    cout << "Please enter the students' information (seperated by Tab): "
      << endl << "id    name            sex yearmonth   day score" << endl;
    for (i = 0; i < n; i++)   //录入学生信息
    {
        cin >> stu[i].id;
        while (cin.peek() == '\t') cin.ignore();   //确保输入 name 前的 Tab 不影响后面输入
        cin.getline(stu[i].name, 30, '\t');   //名字中可以输入空格“ ”，以 Tab 作为终止符
        cin >> stu[i].sex >> stu[i].birthday.year >> stu[i].birthday.month >> stu[i].birthday.day >>
stu[i].score;
        cin.get();   //使换行不影响下一个学生的信息录入
    }
    avr_score = avr(stu, n);
    cout << endl << "The average score is: " << avr_score << endl;
    cout << endl << "Score Rank: " << endl;
    sort_output(stu, n);
    return 0;
}
float avr(struct student *p, int n)   //计算平均分
{
    float avr = 0;
    for (int i = 0; i < n; i++)
    {
        avr += (*p).score / n;
        p++;
    }
    return avr;
}
void sort_output(struct student *p, int n)   //选择排序，按成绩降序排序并输出
{
    student temp;
    int i, j, k;
    for (i = 0; i < n; i++)
    {
        k = i;
        for (j = i + 1; j < n; j++)
            if ((*(p + j)).score > (*(p + k)).score)
                k = j;
        if (k != i)
        {
            temp = *(p+i);
            *(p + i) = *(p + k);
            *(p + k) = temp;
        }
    }
    for (i = 0; i < n; i++)
        cout << "No." << i + 1 << " " << setiosflags(ios::left)
        << setw(18) << (*(p + i)).name << setw(10) << (*(p + i)).score << endl;
}
```

1．矩阵运算 高斯消元：(4x4 矩阵)

```
    for (int k = 0; k <= 3; k++)
    {
        for (int j = k + 1; j <= 4; j++)
            a[k][j] = a[k][j] / a[k][k];
        for (int i = k + 1;i <= 4;i++)
            for (int h = k + 1;h <= 4;h++)
                a[i][h] = a[i][h] - a[i][k] *
            a[k][h];
    }
    double x[4];
    x[3] = a[3][4];
    x[2] = a[2][4] - a[2][3] * x[3];
    x[1] = a[1][4] - a[1][3] * x[3] - a[1][2]
* x[2];
    x[0] = a[0][4] - a[0][3] * x[3] - a[0][2]
* x[2] - a[0][1] * x[1];
```

2．进制转换

```
    for (int k =(自己算一下多少位数);k >=
0;k--)
    //十进制转 m 进制
    b[k] = a % m, a /= m;
```

3．字符串 如何输入?

一个字符串：定义一个字符串数组 gets_s(数组名)

多个字符串(如一句话)：

```
①for (int i=0;i<N(字符串个数);i++)
  {
      p[i]=(char *)malloc(80);
      scanf("%s",p[i]);
  }
②getchar();
  gets_s(word);(其中 word[])
③char *p;
  p = (char*)malloc(20);
  fgets(p, 20, stdin);
  cout << p;
  free(p);
④……
```

伴随矩阵

```
#include<stdio.h>
#include<stdlib.h>
#pragma warning (disable:4996)
void T(int a[], int b[], int n, int i)
{
    int p, q, m;
    m = 0;
    for (p = 0;p < n;p++)
        if (p != i)
        {
            for (q = 0;q < (n - 1);q++)
                b[m*(n - 1) + q] = a[p*n + q + 1];
            m++;
        }
}
int D(int a[], int n)
{
    int i, j, s, *b;
    i = 0;
    j = 0;
    s = 0;
    if (n == 2) s = (a[i*n + j] * a[(i + 1)*n + j +
1]) - (a[i*n + j + 1] * a[(i + 1)*n + j]);
    else
    {
        for (i = 0;i < n;i++)
        {
            b = (int*)malloc((n - 1)*(n - 1) *
sizeof(int));
            T(a, b, n, i);
            if (i % 2 == 0)
                s = s + a[i*n] * D(b, n - 1);
            else s = s - a[i*n] * D(b, n - 1);
            free(b);
        }
    }
    return(s);
}
int C(int a[], int n, int i, int j)
{
    int *b;
    int p, q, m, k, s;
    b = (int*)malloc((n - 1)*(n - 1) * sizeof(int));
    m = 0;
    k = 0;
    for (p = 0;p < n;p++)
        if (p != i)
        {
            k = 0;
            for (q = 0;q < n;q++)
                if (q != j)
                {
                    b[m*(n - 1) + k] = a[p*n + q];
                    k++;
                }
            m++;
```

```
        }
    s = D(b, n - 1);
    return(s);
}
int main()
{
    int i, j, n, *a, *c;
    void T(int a[], int b[], int n, int i);
    int D(int a[], int n);
    int C(int a[], int n, int i, int j);
    printf("please input n=");
    scanf("%d", &n);
    a = (int*)malloc(n*n * sizeof(int));
    c = (int*)malloc(n*n * sizeof(int));
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
        {
            printf("%drow %dcolumn:", i + 1, j + 1);
            scanf("%d", &a[i*n + j]);
        }
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
        {
            if ((i + j) % 2 == 0)
                c[i*n + j] = C(a, n, j, i);
            else c[i*n + j] = -C(a, n, j, i);
        }
    printf("the adj matrix:\n");
    for (i = 0;i < n;i++)
    {
        for (j = 0;j < n;j++)
            printf("%d    ", c[i*n + j]);
        printf("\n");
    }
    free(c);
    free(a);
}
```

矩阵乘积

```
#include<stdio.h>
#include<stdlib.h>
#pragma warning (disable:4996)
void M(int a[], int b[], int c[], int n)
{
    int i, j, k, m;
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
        {
            m = 0;
            for (k = 0;k < n;k++)
                m = m + a[i*n + k] * b[k*n + j];
            c[i*n + j] = m;
        }
}
int main()
{
    int *a, *b, *c;
    int i, j, n;
    void M(int *a, int *b, int *c, int n);
    printf("please input the number n=");
    scanf("%d", &n);
    a = (int*)malloc(n*n * sizeof(int));
    b = (int*)malloc(n*n * sizeof(int));
    c = (int*)malloc(n*n * sizeof(int));
    printf("please input the matrix A:\n");
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            scanf("%d", &a[i*n + j]);
    printf("please input the matrix B:\n");
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            scanf("%d", &b[i*n + j]);
    M(a, b, c, n);
    for (i = 0;i < n;i++)
    {
        for (j = 0;j < n;j++)
            printf("%-5d", c[i*n + j]);
        printf("\n");
    }
    free(a);
    free(b);
    free(c);
}
```

矩阵的行列式求法

```
#include<stdio.h>
#include<stdlib.h>
#pragma warning (disable:4996)
void T(int a[], int b[], int n, int i)
{
    int p, q, m;
    m = 0;
    for (p = 0;p < n;p++)
        if (p != i)
        {
            for (q = 0;q < (n - 1);q++)
                b[m*(n - 1) + q] = a[p*n + q + 1];
            m++;
        }
}
int D(int a[], int n)
{
    int i, j, s, *b;
    i = 0;
    j = 0;
    s = 0;
    if (n == 2) s = (a[i*n + j] * a[(i + 1)*n + j + 1])
- (a[i*n + j + 1] * a[(i + 1)*n + j]);
    else
    {
        for (i = 0;i < n;i++)
        {
            b = (int*)malloc((n - 1)*(n - 1) *
sizeof(int));T(a, b, n, i);
            if (i % 2 == 0)
                s = s + a[i*n] * D(b, n - 1);
            else s = s - a[i*n] * D(b, n - 1);
            free(b);
        }
    }
    return(s);
}
int main()
{
    int i, j, s, n, *a;
    int D(int a[], int n);
    void T(int a[], int b[], int n, int i);
        printf("please input n=");
    scanf("%d", &n);
    a = (int*)malloc(n*n * sizeof(int));
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
```

```
        {
            printf("%drow %dcolumn:", i + 1, j + 1);
            scanf("%d", &a[i*n + j]);
        }
    s = D(a, n);
    printf("D=%d\n", s);
    free(a);
}
```

矩阵的转置

```
#include<stdio.h>
#include<stdlib.h>
#pragma warning (disable:4996)
using namespace std;
void T(int *a, int *b, int n)
{
    int i, j;
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            b[i*n + j] = a[j*n + i];
}
int main()
{
    int *a, *b;
    int n, i, j;
    void T(int *a, int *b, int n);
    printf("please input n=");
    scanf("%d", &n);
    a = (int *)malloc(n*n * sizeof(int));
    b = (int *)malloc(n*n * sizeof(int));
    printf("please input matrix:\n");
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            scanf("%d", &a[i*n + j]);
    T(a, b, n);
    for (i = 0;i < n;i++)
    {
        for (j = 0;j < n;j++)
            printf("%-3d", b[i*n + j]);
        printf("\n");
    }
    free(a);
    free(b);
}
```

选择排序法：

```
int i, j,k, temp,
    a[10] = {32,42,12,3,43,5,3,2,43,2};
    for (i = 0; i <= 9; i++)
    {
        k = i;
        for (j = i + 1; j <= 9; j++)
        {
            if (a[j] > a[k])
            {
                k = j;
            }
        }
        if (k != i)
        {
            temp = a[i];
            a[i] = a[k];
            a[k] = temp;
        }
    }
```

判断素数：

```
    int prime(int m)
    {
        int i, k;
        k = (int)sqrt((float)m);
        for (i = 3;i <= k;i ++)
            if ((m%i)==0)
                return 0;
        return 1;
    }
```

约瑟夫问题：

```
int main()
{
    int num[50], n,k;
    cout << "请输入n:";
    cin >> n;
    int del(int *p, int n);
    k = del(num,n);
    cout << k;
}

int del(int *p, int n)
{
    int sum=0,i,j,N=0;
    for (i = 0; i <= n-1; i++)//赋值
    {
        *(p + i) = 1;
    }
    for (j = 0;; )
    {
        if (*(p + j) == 1)//计数器
        {
            sum++;
        }
        if (sum == 3)
        {
            *(p + j) = 0;
            N++;
            sum = 0;
        }
        j++;
        if (j == n)//从头开始报数
        {
            j = 0;
        }

        if (N == n - 1)break;
    }
    for (i = 0; i <= n-1; i++)//找剩下的最后一个人
    {
        if (*(p + i) == 1)return (i);
    }
}
```

对应数组排列后且输出下标（假设其中无重复数据）：

```
#include<iostream>
using namespace std;
int main()
{
    int *pa[10], a[10], i,j,k, *temp;
    for (i = 0; i <= 9; i++)
    {
        cin >> a[i];
        pa[i] = &a[i];
    }
    for (i = 0; i <= 9; i++)
    {
        k = i;
        for (j= i + 1; j <= 9; j++)
        {
            if (*pa[j] >*pa[k])
            {
                k = j;
            }
        }
        if (k != i)
        {
            temp = pa[i];
            pa[i] = pa[k];
            pa[k] = temp;
        }
    }
    for (i = 0; i <= 9; i++)
    {
        cout << *pa[i] <<'\t';
    }
    cout << endl;
    for(j=0;j<=9;j++)
        for (i = 0; i <= 9; i++)
        {
            if (*pa[j] == a[i])
                cout << i << '\t';
        }
    return 0;
}
```

任意小于十位的整形数组排序：

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int a[10], i=0, j,k,l,temp;
    char c;
    for (;;)
    {
        cin >> a[i];
        if ((c = cin.get()) == '\n')break;
        i++;
    }
    for (j = 0; j <= i-1; j++)
    {
        k = j;
        for (l = j; l <= i; l++)
        {
            if (a[l] > a[k])
            {
                k = l;
            }
        }
        if (k != j)
        {
            temp = a[j];
            a[j] = a[k];
            a[k] = temp;
        }
    }
    for (j = 0; j <= i; j++)cout << a[j] << '\t';
}
```

将 IP 地址转化为去掉其中的.然后转化二进制。

```
#include<iostream>
#include<string>
#include<math.h>
using namespace std;
int main()
{
    char str[16];
    unsigned int x;
    cin >> str;
    unsigned int aton(const char str[]);
    x = aton(str);
    cout << x<<endl;
    if (x == 0)cout << "error!";
    else
    {
        void change(unsigned int N);
        change(x);
    }
    return 0;
}
unsigned int aton(const char str[])
{
    int i, num[4] = {0}, sum = 0;
    unsigned int N;
    for (i = 0;; i++)
    {
        if (str[i] == '\0')break;
        if (str[i] == '.')
        {
            sum++;
            continue;
        }
        if ((str[i] - 48) < 0 || (str[i] - 48) >
9)return 0;
        num[sum] = 10 * num[sum] + (str[i] - 48);

    }
    if (sum != 3)return 0;
    for (i = 0; i <= 3; i++)
    {
        if (num[i] < 0 || num[i]>255)
            return 0;
    }
    N = pow(2, 24)*num[0] + pow(2, 16)*num[1] +
pow(2, 8)*num[2] + num[3];
    return N;
}
void change(unsigned int N)
{
```

```
    int i, a[32] = {0};
    for (i = 0; i <= 31; i++)
    {
        a[i] = N % 2;
        N = N / 2;
    }
    for (i = 31; i >= 0; i--)cout << a[i];
    cout << endl;}
```

对于字符串关于 ASCII 码的排序：

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int i, j,num;
    char temp;
    string s1;
    cin >> s1;
    for (num = 0;; num++)
    {
        if(s1[num] == '\0')
            break;
    }
    num = num - 1;
    for (i = 0;i<=num-1; i++)
    {
        for (j = num-1;j>=0; j--)
        {
                if (s1[j] < s1[j + 1])
                {
                    temp = s1[j];
                    s1[j] = s1[j + 1];
                    s1[j + 1] = temp;
                }

        }
    }
    cout << s1 ;
    return 0;
}
```

去掉重复字符：

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int i, j, k;
    string s1;
    cin >> s1;
    i = s1.length();
    for (j = 1; j <= i; j++)
    {
        for (k = 0; k < j; k++)
        {
            if (s1[k] == s1[j])
            {
                s1[j] = ' ';
            }
        }
    }
    for (j = 0; j <= i - 1; j++)
    {
        if (s1[j] == ' ')continue;
        cout << s1[j];
    }
}
```

矩阵转置：

```
#include<stdio.h>
#include<stdlib.h>
#pragma warning (disable:4996)
using namespace std;
void T(int *a, int *b, int n)
{
    int i, j;
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            b[i*n + j] = a[j*n + i];
}
int main()
{
    int *a, *b;
    int n, i, j;
    void T(int *a, int *b, int n);
    printf("please input n=");
    scanf("%d", &n);
    a = (int *)malloc(n*n * sizeof(int));
    b = (int *)malloc(n*n * sizeof(int));
    printf("please input matrix:\n");
    for (i = 0;i < n;i++)
        for (j = 0;j < n;j++)
            scanf("%d", &a[i*n + j]);
    T(a, b, n);
    for (i = 0;i < n;i++)
    {
        for (j = 0;j < n;j++)
            printf("%-3d", b[i*n + j]);
        printf("\n");
    }
    free(a);
    free(b);
}
```