作业 7

必做题

第一题

按要求的程序代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double f(double);
double S(double,double,int);
int main(){
    int i=1800;
    printf("%.4lf\n",S(-1.0,1.0,i));
    system("pause");
    return 0;
}

/*计算当点函数值*/
double f(double dVariable){
    return exp(-(dVariable*dVariable));
}

/*进行求和*/
double S(double dLower,double dUpper,int iDivision){
    double lfSum=(f(dLower)+f(dUpper))/2;
    double lfWidth=(dUpper-dLower)/iDivision;
    int iRound=1;
    for (;iRound<=iDivision-1;iRound++)
lfSum+=f(dLower+((double)iRound*lfWidth));
    return lfSum*lfWidth;
}
```

运行结果：

```
1.4936
请按任意键继续
```

精度分析：

利用计算器得到积分保留 9 位小数的值为 1.493648266，利用如下程序输出不同精度分割下的结果——

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double f(double);
double S(double,double,int);
int main(){
```

```
    int i;
    for (i=20;i<=8000;i+=20)
    printf("分隔次数%d时的积分值为%.9lf；\n",i,S(-1.0,1.0,i));
    system("pause");
    return 0;
}

/*计算当点函数值*/
double f(double dVariable){
    return exp(-(dVariable*dVariable));
}

/*进行求和*/
double S(double dLower,double dUpper,int iDivision){
    double lfSum=(f(dLower)+f(dUpper))/2;
    double lfWidth=(dUpper-dLower)/iDivision;
    int iRound=1;
    for (;iRound<=iDivision-1;iRound++)
lfSum+=f(dLower+((double)iRound*lfWidth));
    return lfSum*lfWidth;
}
```

运行结果：

分隔次数 20 时的积分值为 1.492421592；
分隔次数 40 时的积分值为 1.493341674；
分隔次数 60 时的积分值为 1.493512009；
分隔次数 80 时的积分值为 1.493571622；
分隔次数 100 时的积分值为 1.493599214；
分隔次数 120 时的积分值为 1.493614202；
分隔次数 140 时的积分值为 1.493623240；
分隔次数 160 时的积分值为 1.493629105；
分隔次数 180 时的积分值为 1.493633126；
分隔次数 200 时的积分值为 1.493636003；
分隔次数 220 时的积分值为 1.493638131；
分隔次数 240 时的积分值为 1.493639750；
分隔次数 260 时的积分值为 1.493641010；
分隔次数 280 时的积分值为 1.493642009；
分隔次数 300 时的积分值为 1.493642816；
分隔次数 320 时的积分值为 1.493643476；
分隔次数 340 时的积分值为 1.493644022；
分隔次数 360 时的积分值为 1.493644481；
分隔次数 380 时的积分值为 1.493644869；
分隔次数 400 时的积分值为 1.493645200；
分隔次数 420 时的积分值为 1.493645485；
分隔次数 440 时的积分值为 1.493645732；

分隔次数 460 时的积分值为 1.493645948；
分隔次数 480 时的积分值为 1.493646137；
分隔次数 500 时的积分值为 1.493646304；
分隔次数 520 时的积分值为 1.493646452；
分隔次数 540 时的积分值为 1.493646584；
分隔次数 560 时的积分值为 1.493646702；
分隔次数 580 时的积分值为 1.493646808；
分隔次数 600 时的积分值为 1.493646903；
分隔次数 620 时的积分值为 1.493646990；
分隔次数 640 时的积分值为 1.493647068；
分隔次数 660 时的积分值为 1.493647140；
分隔次数 680 时的积分值为 1.493647205；
分隔次数 700 时的积分值为 1.493647265；
分隔次数 720 时的积分值为 1.493647319；
分隔次数 740 时的积分值为 1.493647370；
分隔次数 760 时的积分值为 1.493647416；
分隔次数 780 时的积分值为 1.493647459；
分隔次数 800 时的积分值为 1.493647499；
分隔次数 820 时的积分值为 1.493647536；
分隔次数 840 时的积分值为 1.493647570；
分隔次数 860 时的积分值为 1.493647602；
分隔次数 880 时的积分值为 1.493647632；
分隔次数 900 时的积分值为 1.493647660；
分隔次数 920 时的积分值为 1.493647686；
分隔次数 940 时的积分值为 1.493647711；
分隔次数 960 时的积分值为 1.493647733；
分隔次数 980 时的积分值为 1.493647755；
分隔次数 1000 时的积分值为 1.493647775；
分隔次数 1020 时的积分值为 1.493647794；
分隔次数 1040 时的积分值为 1.493647812；
分隔次数 1060 时的积分值为 1.493647829；
分隔次数 1080 时的积分值为 1.493647845；
分隔次数 1100 时的积分值为 1.493647860；
分隔次数 1120 时的积分值为 1.493647875；
分隔次数 1140 时的积分值为 1.493647888；
分隔次数 1160 时的积分值为 1.493647901；
分隔次数 1180 时的积分值为 1.493647913；
分隔次数 1200 时的积分值为 1.493647925；
分隔次数 1220 时的积分值为 1.493647936；
分隔次数 1240 时的积分值为 1.493647947；
分隔次数 1260 时的积分值为 1.493647957；
分隔次数 1280 时的积分值为 1.493647966；
分隔次数 1300 时的积分值为 1.493647975；
分隔次数 1320 时的积分值为 1.493647984；

分隔次数 1340 时的积分值为 1.493647992；
分隔次数 1360 时的积分值为 1.493648000；
分隔次数 1380 时的积分值为 1.493648008；
分隔次数 1400 时的积分值为 1.493648015；
分隔次数 1420 时的积分值为 1.493648022；
分隔次数 1440 时的积分值为 1.493648029；
分隔次数 1460 时的积分值为 1.493648036；
分隔次数 1480 时的积分值为 1.493648042；
分隔次数 1500 时的积分值为 1.493648048；
分隔次数 1520 时的积分值为 1.493648053；
分隔次数 1540 时的积分值为 1.493648059；
分隔次数 1560 时的积分值为 1.493648064；
分隔次数 1580 时的积分值为 1.493648069；
分隔次数 1600 时的积分值为 1.493648074；
分隔次数 1620 时的积分值为 1.493648079；
分隔次数 1640 时的积分值为 1.493648083；
分隔次数 1660 时的积分值为 1.493648088；
分隔次数 1680 时的积分值为 1.493648092；
分隔次数 1700 时的积分值为 1.493648096；
分隔次数 1720 时的积分值为 1.493648100；
分隔次数 1740 时的积分值为 1.493648104；
分隔次数 1760 时的积分值为 1.493648107；
分隔次数 1780 时的积分值为 1.493648111；
分隔次数 1800 时的积分值为 1.493648114；
分隔次数 1820 时的积分值为 1.493648118；
分隔次数 1840 时的积分值为 1.493648121；
分隔次数 1860 时的积分值为 1.493648124；
分隔次数 1880 时的积分值为 1.493648127；
分隔次数 1900 时的积分值为 1.493648130；
分隔次数 1920 时的积分值为 1.493648133；
分隔次数 1940 时的积分值为 1.493648135；
分隔次数 1960 时的积分值为 1.493648138；
分隔次数 1980 时的积分值为 1.493648141；
分隔次数 2000 时的积分值为 1.493648143；
分隔次数 2020 时的积分值为 1.493648145；
分隔次数 2040 时的积分值为 1.493648148；
分隔次数 2060 时的积分值为 1.493648150；
分隔次数 2080 时的积分值为 1.493648152；
分隔次数 2100 时的积分值为 1.493648154；
分隔次数 2120 时的积分值为 1.493648156；
分隔次数 2140 时的积分值为 1.493648159；
分隔次数 2160 时的积分值为 1.493648160；
分隔次数 2180 时的积分值为 1.493648162；
分隔次数 2200 时的积分值为 1.493648164；

分隔次数 2220 时的积分值为 1.493648166；
分隔次数 2240 时的积分值为 1.493648168；
分隔次数 2260 时的积分值为 1.493648170；
分隔次数 2280 时的积分值为 1.493648171；
分隔次数 2300 时的积分值为 1.493648173；
分隔次数 2320 时的积分值为 1.493648174；
分隔次数 2340 时的积分值为 1.493648176；
分隔次数 2360 时的积分值为 1.493648178；
分隔次数 2380 时的积分值为 1.493648179；
分隔次数 2400 时的积分值为 1.493648180；
分隔次数 2420 时的积分值为 1.493648182；
分隔次数 2440 时的积分值为 1.493648183；
分隔次数 2460 时的积分值为 1.493648185；
分隔次数 2480 时的积分值为 1.493648186；
分隔次数 2500 时的积分值为 1.493648187；
分隔次数 2520 时的积分值为 1.493648188；
分隔次数 2540 时的积分值为 1.493648190；
分隔次数 2560 时的积分值为 1.493648191；
分隔次数 2580 时的积分值为 1.493648192；
分隔次数 2600 时的积分值为 1.493648193；
分隔次数 2620 时的积分值为 1.493648194；
分隔次数 2640 时的积分值为 1.493648195；
分隔次数 2660 时的积分值为 1.493648196；
分隔次数 2680 时的积分值为 1.493648197；
分隔次数 2700 时的积分值为 1.493648198；
分隔次数 2720 时的积分值为 1.493648199；
分隔次数 2740 时的积分值为 1.493648200；
分隔次数 2760 时的积分值为 1.493648201；
分隔次数 2780 时的积分值为 1.493648202；
分隔次数 2800 时的积分值为 1.493648203；
分隔次数 2820 时的积分值为 1.493648204；
分隔次数 2840 时的积分值为 1.493648205；

可见 m>=100 时的精度就大致能够保证 1e-4 的计算精度要求。

利用这个积分算法，可以写出一个任意输入函数输出导数或积分的程序：

```
/*文件包含*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*模式宏定义*/
#define CALCULATE 1
#define TANGENT 2
```

```
#define SUM 3

/*函数声明*/
double fMain(double);
double f(double);
double S(double,double,int);
double Dev(double,double);
double dRead(double);
char cJudgeNum(int);
char cJudgeNumAdvanced(int);
char cPrintError();
double dA(double,double);/*addition*/
double dS(double,double);/*substraction*/
double dM(double,double);/*multiplication*/
double dD(double,double);/*divition*/
double dP(double,double);/*power*/
char cMode();

/*常数定义*/
const double e=2.718281828;
const double pi=3.1415926535898;

/*全局变量定义*/
char s[1000];
int iPosition=0;
char cCheck=0;

/*主函数*/
int main(){
    char cMInput=0;
    double dMResult=1;
    int iRMain=0;
    char cGet=100;
    int iMainCalculateRound=0;
    double dPositionDev=1.0;
    double dLow=0;
    double dUp=0;
    /*获取表达式*/
    printf("本程序默认的运算优先级和数学相同，但是与字母紧密结合的常数优先级最
高。\n");
    printf("请输入一个表达式：");
    for (;cGet!=10;iRMain++) {
        cGet=getchar();
        s[iRMain]=cGet;
```

```
    }
    /*选择输出模式*/
    cMInput=cMode();
    if (cMInput==CALCULATE){/*直接计算模式*/
        for (;iMainCalculateRound<1000;iMainCalculateRound++){
            if (s[iMainCalculateRound]=='x') cPrintError();
        }
        dMResult=f(2.0);
        if (cCheck==0) printf("表达式计算结果为：%.7lf。\n",dMResult);
    }
    else if (cMInput==TANGENT){/*求导模式*/
        printf("请输入对函数求导的x值（用含小数点的小数表示）：");
        scanf("%lf",&dPositionDev);
        dMResult=Dev(dPositionDev,0.00001);
        if (cCheck==0) printf("在x=%.7lf处求导的近似结果为%.7lf。
\n",dPositionDev,dMResult);
    }
    else if (cMInput==SUM){/*求定积分模式*/
        printf("请输入积分下限和积分上限，用空格隔开：");
        scanf("%lf %lf",&dLow,&dUp);
        dMResult=S(dLow,dUp,8000);
        printf("%lf,%lf\n",dLow,dUp);
        if (cCheck==0) printf("积分近似值为%.7lf；\n",dMResult);
    }
    system("pause");
    return 0;
}

/*作为一个完整的函数*/
double fMain(double dVariable){
    iPosition=0;
    return f(dVariable);
}

/*作为函数括号中的部分*/
double f(double dVariable){
    double dBuffer=dRead(dVariable);/*录入第一个数*/
    while (1){
        if (cJudgeNum(iPosition)){/*输入有问题，读完数还有数*/
            printf("%d\t",iPosition);
            cPrintError();
            return 1;
        }
        else if (s[iPosition]=='('){/*进入括号*/
```

```
            iPosition++;
            dBuffer=f(dVariable);
        }
        switch(s[iPosition++]){/*进行下一步计算或退出*/
        case ')': return dBuffer;break;
        case 10: iPosition--; return dBuffer;break;
        case '+': dBuffer=dA(dBuffer,dVariable);break;
        case '-': dBuffer=dS(dBuffer,dVariable);break;
        case '*': dBuffer=dM(dBuffer,dVariable);break;
        case '/': dBuffer=dD(dBuffer,dVariable);break;
        case '^': dBuffer=dP(dBuffer,dVariable);
        }
    }
}

/*进行积分求和*/
double S(double dLower,double dUpper,int iDivision){
    double lfSum=(fMain(dLower)+fMain(dUpper))/2;
    double lfWidth=(dUpper-dLower)/iDivision;
    int iRound=1;
    for (;iRound<=iDivision-1;iRound++)
lfSum+=fMain(dLower+((double)iRound*lfWidth));
    return lfSum*lfWidth;
}

/*在某点处求导*/
double Dev(double dPos,double dDelta){
    return (fMain(dPos+dDelta)-fMain(dPos))/dDelta;
}

/*对数据进行录入*/
double dRead(double dX){
    double dResult=0;
    /*上来就是字母的情形*/
    if (cJudgeNum(iPosition)&&(!cJudgeNumAdvanced(iPosition))){
        if (s[iPosition]=='e') {
            iPosition++;
            return e;
        }
        else if (s[iPosition]=='p'){
            iPosition+=2;
            return pi;
        }
        else if (s[iPosition]=='x'){
```

```
            iPosition++;
            return dX;
        }
        else if (s[iPosition]=='l'){
            iPosition+=2;
            return log(f(dX));
        }
        else if (s[iPosition]=='s'){
            iPosition+=3;
            return sin(f(dX));
        }
        else if (s[iPosition]=='c'){
            iPosition+=3;
            return cos(f(dX));
        }
        else if (s[iPosition]=='t'){
            iPosition+=3;
            return tan(f(dX));
        }
        else if (s[iPosition]=='a'){/*反三角函数第一位相同*/
            iPosition+=3;
            if (cJudgeNum(iPosition)){
                switch (s[iPosition]){
                case 's': iPosition+=3; return asin(f(dX)); break;
                case 'c': iPosition+=3; return acos(f(dX)); break;
                case 't': iPosition+=3; return atan(f(dX)); break;
                }
            }
            else{
                cPrintError();
                return 1;
            }
        }
    }
    /*整数部分*/
    while (cJudgeNumAdvanced(iPosition)){
        dResult*=10;
        dResult+=s[iPosition++]-'0';
    }
    /*小数部分*/
    if (s[iPosition]=='.'){
        unsigned short usExpoTen=1;
        if (!cJudgeNumAdvanced(iPosition+1)){
            cPrintError();
```

```
            return 1;
        }
        do {
            dResult+=(double)(s[++iPosition]-
'0')/pow((double)10,(int)usExpoTen++);
        } while (cJudgeNumAdvanced(iPosition+1));
        iPosition++;
    }
    /*数乘了字母*/
    if (cJudgeNum(iPosition)){
        switch(s[iPosition]){
        case 'e': dResult*=e; iPosition++; break;
        case 'p': dResult*=pi; iPosition+=2; break;
        case 'x': dResult*=dX; iPosition++; break;
        case 'l': iPosition+=2; dResult*=log(f(dX)); break;
        case 's': iPosition+=3; dResult*=sin(f(dX)); break;
        case 'c': iPosition+=3; dResult*=cos(f(dX)); break;
        case 't': iPosition+=3; dResult*=tan(f(dX)); break;
        case 'a': /*反三角函数第一位相同*/
            iPosition+=3;
            if (cJudgeNum(iPosition)){
                switch (s[iPosition]){
                case 's': iPosition+=3; dResult*=asin(f(dX)); break;
                case 'c': iPosition+=3; dResult*=acos(f(dX)); break;
                case 't': iPosition+=3; dResult*=atan(f(dX)); break;
                }
            }
            else{
                cPrintError();
                return 1;
            }
        }
    }
    return dResult;
}

/*判断是需要录入的数据还是运算符*/
char cJudgeNum(int iPosit){
    return ((s[iPosit]>='0' && s[iPosit]<='9') ||
        s[iPosit]=='e' ||
        (s[iPosit]=='p' && s[iPosit+1]=='i') ||
        s[iPosit]=='x' ||
        (s[iPosit]=='s' && s[iPosit+1]=='i' && s[iPosit+2]=='n') ||
        (s[iPosit]=='c' && s[iPosit+1]=='o' && s[iPosit+2]=='s') ||
```

```
        (s[iPosit]=='t' && s[iPosit+1]=='a' && s[iPosit+2]=='n') ||
        (s[iPosit]=='a' && s[iPosit+1]=='r' && s[iPosit+2]=='c') ||
        (s[iPosit]=='l' && s[iPosit+1]=='n'));
}

/*判断是不是数字*/
char cJudgeNumAdvanced(int iPosit){
    return (s[iPosit]>='0' && s[iPosit]<='9');
}

/*输入有误时的提醒*/
char cPrintError(){
    printf("输入错误，请退出程序重新输入。\n");
    cCheck=1;
    return 0;
}

/*乘方*/
double dP(double dLeft,double dX){
    double dRight=dRead(dX);
        if (cJudgeNum(iPosition)){
            cPrintError();
            return 1;
        }
        else if (s[iPosition]=='('){
            iPosition++;
            dRight=f(dX);
        }
        switch(s[iPosition]){
        case ')':
        case 10:
        case '+':
        case '-':
        case '*':
        case '/': return pow(dLeft,dRight);break;
        case '^': iPosition++;return pow(dLeft,dP(dRight,dX));
        }
}

/*乘法*/
double dM(double dLeft,double dX){
    double dRight=dRead(dX);
        if (cJudgeNum(iPosition)){
            cPrintError();
```

```
            return 1;
        }
        else if (s[iPosition]=='('){
            iPosition++;
            dRight=f(dX);
        }
        switch(s[iPosition]){
        case ')': return dLeft*dRight;break;
        case 10: return dLeft*dRight;break;
        case '+': return dLeft*dRight;break;
        case '-': return dLeft*dRight;break;
        case '*': iPosition++;return dM(dLeft*dRight,dX);break;
        case '/': iPosition++;return dD(dLeft*dRight,dX);break;
        case '^': iPosition++;return dLeft*dP(dRight,dX);
        }
}

/*除法*/
double dD(double dLeft,double dX){
    double dRight=dRead(dX);
        if (cJudgeNum(iPosition)){
            cPrintError();
            return 1;
        }
        else if (s[iPosition]=='('){
            iPosition++;
            dRight=f(dX);
        }
        if (fabs(dRight-0.0)<=1E-9){
            cPrintError();
            return 1;
        }
        switch(s[iPosition]){
        case ')': return dLeft/dRight;break;
        case 10: return dLeft/dRight;break;
        case '+': return dLeft/dRight;break;
        case '-': return dLeft/dRight;break;
        case '*': iPosition++;return dM(dLeft/dRight,dX);break;
        case '/': iPosition++;return dD(dLeft/dRight,dX);break;
        case '^': iPosition++;return dLeft/dP(dRight,dX);
        }
}

/*加法*/
```

```
double dA(double dLeft,double dX){
    double dRight=dRead(dX);
        if (cJudgeNum(iPosition)){
            cPrintError();
            return 1;
        }
        else if (s[iPosition]=='('){
            iPosition++;
            dRight=f(dX);
        }
        switch(s[iPosition]){
        case ')': return dLeft+dRight;break;
        case 10: return dLeft+dRight;break;
        case '+': iPosition++;return dA(dLeft+dRight,dX);break;
        case '-': iPosition++;return dS(dLeft+dRight,dX);break;
        case '*': iPosition++;return dLeft+dM(dRight,dX);
        case '/': iPosition++;return dLeft+dD(dRight,dX);
        case '^': iPosition++;return dLeft+dP(dRight,dX);
        }
}

/*减法*/
double dS(double dLeft,double dX){
    double dRight=dRead(dX);
        if (cJudgeNum(iPosition)){
            cPrintError();
            return 1;
        }
        else if (s[iPosition]=='('){
            iPosition++;
            dRight=f(dX);
        }
        switch(s[iPosition]){
        case ')': return dLeft-dRight;break;
        case 10: return dLeft-dRight;break;
        case '+': iPosition++;return dA(dLeft-dRight,dX);break;
        case '-': iPosition++;return dS(dLeft-dRight,dX);break;
        case '*': iPosition++;return dLeft-dM(dRight,dX);
        case '/': iPosition++;return dLeft-dD(dRight,dX);
        case '^': iPosition++;return dLeft-dP(dRight,dX);
        }
}

/*判断输出内容*/
```

```
char cMode(){
    unsigned short usInput=0;
    printf("请选择输出模式——（1：直接计算，2：求导，3：求定积分）\t");
    scanf("%hu",&usInput);
    switch(usInput){
    case CALCULATE:return CALCULATE;break;
    case TANGENT:return TANGENT;break;
    case SUM:return SUM;break;
    default:cPrintError();return 0;
    }
}
```

本题运行结果：

```
本程序默认的运算优先级和数学相同，但是与字母紧密结合的常数优先级最高。
请输入一个表达式：e^(-x^2)
请选择输出模式--（1：直接计算，2：求导，3：求定积分）    3
请输入积分下限和积分上限，用空格隔开：-1 1
-1.000000,1.000000
积分近似值为1.4936483;
请按任意键继续. . .
```

第二题

程序代码：

```
#include <stdio.h>
#include <stdlib.h>
int fStraw(char,char,char);
int main(){
    char cStanding,cLying,cOld;
    for (cOld=3;cOld<100;cOld+=3){
        for (cStanding=1;(cStanding+cOld)<100;cStanding++){
            cLying=100-cOld-cStanding;
            if (cLying>0 && fStraw(cStanding,cLying,cOld)==100) printf("站着的小水牛%d头，躺着的小水牛%d头，老水牛%d头。\n",cStanding,cLying,cOld);
        }
    }
    system ("pause");
    return 0;
}

int fStraw(char cStand,char cLie, char cElder){
    int iStraw=0;
    iStraw+=5*cStand;/*站着的小水牛五捆草*/
    iStraw+=3*cLie;/*躺着的小水牛三捆草*/
    iStraw+=cElder/3;/*老水牛三头一捆草*/
    return iStraw;
```

```
}
```

运行结果：

```
站着的小水牛4头，躺着的小水牛18头，老水牛78头。
站着的小水牛8头，躺着的小水牛11头，老水牛81头。
站着的小水牛12头，躺着的小水牛4头，老水牛84头。
请按任意键继续. . .
```

选做题

第一题

程序代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(){
    double dLog=0;
    unsigned short usRound=1;
    for (;usRound<=1000;usRound++){
        dLog+=log10((double)usRound);
    }
    printf("1000的阶乘的位数为%d。",(int)dLog+1);
    system("pause");
    return 0;
}
```

运行结果：

```
1000的阶乘的位数为2568。请按任意键继续. . .
```

第五题

程序代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
char cCheck(char,char);
char cFill(char,char);
char cJudgeMatrix();
char cJudge15(char,char,char);
char cPrint();
char cHuanFang[9];
int main(){
    char cRoundMain=0;
    for (cRoundMain=1;cRoundMain<=9;cRoundMain++){
        cFill((char)0,cRoundMain);
```

```
    }
    system("pause");
    return 0;
}

/*检查一个数字是否在之前的位置里出现过*/
char cCheck(char cCandidate,char cPosition){
    char cResult=0;
    char cRoundCheck=0;
    for (;cRoundCheck<cPosition;cRoundCheck++){
        cResult+=(cCandidate==cHuanFang[cRoundCheck]);
    }
    return cResult;
}

/*填充幻方的递归函数*/
char cFill(char cPositionFilled,char cNumberFilled){
    char cCandidateNumber;
    cHuanFang[cPositionFilled]=cNumberFilled;
    if (cPositionFilled!=8){
        for (cCandidateNumber=1;cCandidateNumber<=9;cCandidateNumber++){
            if (cCheck(cCandidateNumber,cPositionFilled+1)==0)
cFill(cPositionFilled+1,cCandidateNumber);
        }
    }
    else cJudgeMatrix();
}

/*判断方阵是否为幻方的函数*/
char cJudgeMatrix(){
    char cJudgeSum=0;
    char cProcess=0;
    /*行检查*/
    for (;cProcess<=2;cProcess++){

    cJudgeSum+=cJudge15(cHuanFang[cProcess*3],cHuanFang[cProcess*3+1],cHuanFang
[cProcess*3+2]);
    }
    /*列检查*/
    for (cProcess=0;cProcess<=2;cProcess++){

    cJudgeSum+=cJudge15(cHuanFang[cProcess],cHuanFang[cProcess+3],cHuanFang[cPr
ocess+6]);
    }
```

```
    /*对角线检查*/
    cJudgeSum+=cJudge15(cHuanFang[0],cHuanFang[4],cHuanFang[8]);
    cJudgeSum+=cJudge15(cHuanFang[2],cHuanFang[4],cHuanFang[6]);
    /*输出*/
    if (cJudgeSum==0) cPrint();
    return 0;
}

/*判断三个数之和是否为15的函数*/
char cJudge15(char cOne,char cTwo,char cThree){
    return ((cOne+cTwo+cThree)!=15);
}

/*打印幻方的函数*/
char cPrint(){
    char cRow,cColumn;
    printf("\n如下幻方符合条件：\n");
    for (cRow=0;cRow<=2;cRow++){
        for (cColumn=0;cColumn<=2;cColumn++){
            printf("%d\t",cHuanFang[cRow*3+cColumn]);
        }
        printf("\n");
    }
}
```

运行结果：

```
如下幻方符合条件：
2       7       6
9       5       1
4       3       8

如下幻方符合条件：
2       9       4
7       5       3
6       1       8

如下幻方符合条件：
4       3       8
9       5       1
2       7       6

如下幻方符合条件：
4       9       2
```

3 5 7
8 1 6

如下幻方符合条件：
6 1 8
7 5 3
2 9 4

如下幻方符合条件：
6 7 2
1 5 9
8 3 4

如下幻方符合条件：
8 1 6
3 5 7
4 9 2

如下幻方符合条件：
8 3 4
1 5 9
6 7 2