误差理论与测量平差上机指导书

辽宁工程技术大学

测绘与地理科学学院 测绘工程系

目录

Visual C++ 6.0 开发平台简介 1
MFC 概述 1
实验 1 矩阵加法与乘法运算 3
实验 2 矩阵转置与求逆运算 6
实验 3 误差椭圆元素计算 13
实验 4 水准网间接平差程序设计 15

Visual C++ 6.0 开发平台简介

Visual C++提供了一个支持可视化编程的集成开发环境：Visual Studio(又名 Developer Studio)。Developer Studio 是一个通用的应用程序集成开发环境，它不仅支持 Visual C++，还支持 Visual Basic,Visual J++,Visual InterDev 等 Microsoft 系列开发工具。Developer Studio 包含了一个文本编辑器、资源编辑器、工程编译工具、一个增量连接器、源代码浏览器、集成调试工具，以及一套联机文档。使用 Developer Studio，可以完成创建、调试、修改应用程序等的各种操作。

Developer Studio 采用标准的多窗口 Windows 用户界面，并增加了一些新特性，使得开发环境更易于使用，用户很容易学会它的使用方法。

由于 Developer Studio 是一个可视化的开发工具，在介绍 Developer Studio 的各个组成部分之前，首先了解一下可视化编程的概念。可视化技术是当前发展迅速并引人注目的技术之一，它的特点是把原来抽象的数字、表格、功能逻辑等用直观的图形、图象的形式表现出来。可视化编程是它的重要应用之一。所谓可视化编程，就是指：在软件开发过程中，用直观的具有一定含义的图标按钮、图形化的对象取代原来手工的抽象的编辑、运行、浏览操作，软件开发过程表现为鼠标点击按钮和拖放图形化的对象以及指定对象的属性、行为的过程。这种可视化的编程方法易学易用，而且大大提高了工作效率。

Visual C++的集成开发环境 Developer Studio 提供了大量的实用工具以支持可视化编程特性，它们包括：项目工作区、ClassWizard、AppWizard、WizardBar、Component Gallery 等。

MFC 概述

是一个编程框架。 中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上来建立 下的应用程序,这是一种相对 来说更为简单的方法。

因为总体上，　　C框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。　　提供了相应的工C具+来完成这个工作：

　　可以用来生成初步的框架文件（代码和资源等）；资源编辑器用于帮助直观地设计用户接口；C　　用来协助添加代码到框架文件；最后，编译，则通过类库实现了应用程序特定的逻辑。

封装

　　构成　　框架的是　　类库。　　类库是 C++类库。这些类或者封装了

　　应用程序编程接口，或者封装了应用程序的概念，或者封装了　　特性，或者封装了　　和C　　数据访问的功能，等等。

继承

　　首先，　　C抽象出众多类的共同特性，设计出一些基类作为实现其他类的基础。这些类中，最重要的类是 C　　和 CC　　。C　　是　　C的根类，绝大多数　　类是其派生的，包括 CC　　。C　　实现了一些重要的特性，包括动态类信息、动态创建、对象序列化、对程序调试的支持，等等。所有从 C　　派生的类都将具备或者可以具备 C　　所拥有的特性。CC　　通过封装一些属性和方法，提供了消息处理的架构。　　C中，任何可以处理消息的类都从 CC　　派生。

　　针对每种不同的对象，　　C都设计了一组类对这些对象进行封装，每一组类都有一个基类，从基类派生出众多更具体的类。这些对象包括以下种类：窗口对象，基类是 C　　；应用程序对象，基类是 C　　；文档对象，基类是 C　　，等等。程序员将结合自己的实际，从适当的　　类中派生出自己的类，实现特定的功能，达到自己的编程目的。

虚拟函数和动态约束

　　以“C++”为基础，自然支持虚拟函数和动态约束。但是作为一个编程框架，有一个问题必须解决：如果仅仅通过虚拟函数来支持动态约束，必然导致虚拟函数表过于臃肿，消耗内存，效率低下。例如，C　　封装窗口对象时，每一条　　消息对应一个成员函数，这些成员函数为派生类所继承。如果这些函数都设计成虚拟函数，由于数量太多，实现起来不现实。于是，　　C建立了消息映射机制，以一种富有效率、便于使用的手段解决消息处理函数的动态约束问题。

　　这样，通过虚拟函数和消息映射，　　C类提供了丰富的编程接口。程序员继承基类的同时，把自己实现的虚拟函数和消息处理函数嵌入　　C的编程

框架。　　编程框架将在适当的时候、适当的地方来调用程序的代码。

的宏观框架体系

如前所述，　　实现了对应用程序概念的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。这样封装的结果对程序员来说，是一套开发模板（或者说模式）。针对不同的应用和目的，程序员采用不同的模板。例如，　　应用程序的模板，　　应用程序的模板，规则　　应用程序的模板，扩展　　应用程序的模板，　　应用程序的模板，等等。这些模板都采用了以文档 视为中心的思想，每一个模板都包含一组特定的类。

为了支持对应用程序概念的封装，　　内部必须作大量的工作。例如，为了实现消息映射机制，　　编程框架必须要保证首先得到消息，然后按既定的方法进行处理。又如，为了实现对　　编程的支持和多线程编程的支持，

内部使用了特别的处理方法，使用模块状态、线程状态等来管理一些重要信息。虽然，这些内部处理对程序员来说是透明的，但是，懂得和理解 内部机制有助于写出功能灵活而强大的程序。

总之，　　封装了　　，　　，　　等底层函数的功能，并提供更高一层的接口，简化了　　编程。同时，　　支持对底层 的直接调用。　　提供了一个　　应用程序开发模式，对程序的控制主要是由　　框架完成的，而且　　也完成了大部分的功能，预定义或实现了许多事件和消息处理，等等。框架或者由其本身处理事件，不依赖程序员的代码；或者调用程序员的代码来处理应用程序特定的事件。

是　　类库，程序员就是通过使用、继承和扩展适当的类来实现特定的目的。例如，继承时，应用程序特定的事件由程序员的派生类来处理，不感兴趣的由基类处理。实现这种功能的基础是　　对继承的支持，对虚拟函数的支持，以及　　实现的消息映射机制。

实验 1 矩阵加法与乘法运算

一、　实验名称：矩阵加法与乘法运算。

二、　实验目的和任务：掌握矩阵加法与乘法通用程序的编写。

三、 实验要求：

1 每人独立编写出矩阵加法与乘法的程序，并上机调试通过；

2 采用 VC++6.0 开发平台，C 或者 C++语言编写程序；

3 写出矩阵运算的结果。

四、 实验内容：

1 矩阵加法的示例函数(C 语言)

```
void JZjiafa(double a[15][15],double b[15][15],double c[15][15],int m,int n)
{
    for (int i=0;i<=m-1;i++)
        for(int j=0;j<=n-1;j++)
        {
            c[i][j]=a[i][j]+b[i][j];
        }
    return;
}
```

2 矩阵乘法的示例程序(C 语言)

```
#include "stdafx.h"
void matrixMultiply(double a[14][15],double b[15][13], double c[14][13],long m,long n,long k)
{
    for (long i = 0; i<= m-1; i++)
    {
        for (long j=0; j<=k-1; j++)
        {
            c[i][j] =0.0;
            for (long q=0; q<=n-1;q++)
            {
                c[i][j] = c[i][j] + a[i][q] * b[q][j];
            }
        }
}
```

```
        return;
    }

    int main(int argc, char* argv[])
    {
        long n,m,k,i,j;
        double a[14][15],c[14][13],b[15][13];
        FILE *stream;
        stream = fopen("矩阵输入.txt","r");
        fscanf(stream,"%ld %ld",&n,&m);
        for (i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {
                fscanf(stream,"%lf",&a[i][j]);
            }
        }
        fscanf(stream,"%ld %ld",&m,&k);
        for(i=0;i<m;i++)
        {
            for(j=0;j<k;j++)
            {
                fscanf(stream,"%lf",&b[i][j]);
            }
        }
        fclose(stream);
        matrixMultiply(a,b,c,4,5,3);
        stream = fopen("矩阵计算结果.txt","w");
        for (i=0;i<=3;i++)
        {
            for(j=0;j<=2;j++)
                fprintf(stream,"%16.7e    ",c[i][j]);
                fprintf(stream,"\n");
```

```
    }
    fprintf(stream,"\n");
    fclose(stream);
    return 0;
}
```

实验 2 矩阵转置与求逆运算

一、 实验名称：矩阵转置与求逆运算。

二、 实验目的和任务：掌握矩阵转置的编写，会调用矩阵求逆函数。

三、 实验要求：

1 每人独立编写出矩阵转置的程序，并上机调试通过；

2 每人独立完成矩阵求逆函数的调用，并调试通过；

3 采用 VC++6.0 开发平台，C 或者 C++语言编写程序；

4 写出矩阵运算的结果。

四、 实验内容：

i. 矩阵的转置示例函数(C 语言)

```
double JZzhuanzhi(double a[15][15], double b[15][15], int m,int n)
 {
      {
       for(int i=0;i<m;i++)
              for(int j=0;j<n;j++)
                   b[j][i]=a[i][j];
      }
 return 0.0;
 }
```

ii. 矩阵求逆的示例函数(C 语言)

```
int invGJ(double **a,int n)
{
    int *is,*js,i,j,k,l,u,v;
    double d,p;
    is=(int *)malloc(n*sizeof(int));
    js=(int *)malloc(n*sizeof(int));
    for(k=0;k<=n-1;k++)
    {
        d=0.0;
    for(i=k;i<=n-1;i++)
    for(j=k;j<=n-1;j++)
    {
        l=i*n+j;p=fabs(a[i][j]);
        if(p>d)
        {d=p;is[k]=i;js[k]=j;}
    }
 if(d+1.0==1.0)
{
    free(is);free(js);printf("error not inv\n");
    return (0);
}
if(is[k]!=k)
for(j=0;j<=n-1;j++)
{
    u=k*n+j;v=is[k]*n+j;
    p=a[k][j];a[k][j]=a[is[k]][j];a[is[k]][j]=p;
}

if(js[k]!=k)
   for(i=0;i<=n-1;i++)
   {
       u=i*n+k;v=i*n+js[k];
```

```
                p=a[i][k];a[i][k]=a[i][js[k]];a[i][js[k]]=p;
            }
l=k*n+k;
a[k][k]=1.0/a[k][k];
for(j=0;j<=n-1;j++)
      if(j!=k)
      {
            u=k*n+j;a[k][j]=a[k][j]*a[k][k];
      }
  for(i=0;i<=n-1;i++)
     if(i!=k)
    for(j=0;j<=n-1;j++)
         if(j!=k)
         {
               u=i*n+j;
               a[i][j]=a[i][j]-a[i][k]*a[k][j];
         }
  for(i=0;i<=n-1;i++)
       if(i!=k)
       {
            u=i*n+k;a[i][k]=-a[i][k]*a[k][k];
       }
}
for(k=n-1;k>=0;k--)
{
  if(js[k]!=k)
          for(j=0;j<=n-1;j++)
          {
               u=k*n+j;v=js[k]*n+j;
               p=a[k][j];a[k][j]=a[js[k]][j];a[js[k]][j]=p;
          }
  if(is[k]!=k)
       for(i=0;i<=n-1;i++)
```

```
            {
                u=i*n+k;v=i*n+is[k];
                p=a[i][k];a[i][k]=a[i][is[k]];a[i][is[k]]=p;
            }
        }
        free(is);free(js);
        return (1);
} int invGJ(double **a,int n)
{
        int *is,*js,i,j,k,l,u,v;
        double d,p;
        is=(int *)malloc(n*sizeof(int));
        js=(int *)malloc(n*sizeof(int));
        for(k=0;k<=n-1;k++)
        {
            d=0.0;
            for(i=k;i<=n-1;i++)
            for(j=k;j<=n-1;j++)
            {
                l=i*n+j;p=fabs(a[i][j]);
                if(p>d)
                {d=p;is[k]=i;js[k]=j;}
            }
         if(d+1.0==1.0)
        {
            free(is);free(js);printf("error not inv\n");
            return (0);
        }
        if(is[k]!=k)
            for(j=0;j<=n-1;j++)
            {
                u=k*n+j;v=is[k]*n+j;
                p=a[k][j];a[k][j]=a[is[k]][j];a[is[k]][j]=p;
```

```
        }

        if(js[k]!=k)
           for(i=0;i<=n-1;i++)
           {
                u=i*n+k;v=i*n+js[k];
                p=a[i][k];a[i][k]=a[i][js[k]];a[i][js[k]]=p;
           }
    l=k*n+k;
    a[k][k]=1.0/a[k][k];
    for(j=0;j<=n-1;j++)
        if(j!=k)
        {
             u=k*n+j;a[k][j]=a[k][j]*a[k][k];
        }
     for(i=0;i<=n-1;i++)
        if(i!=k)
       for(j=0;j<=n-1;j++)
           if(j!=k)
           {
                u=i*n+j;
                a[i][j]=a[i][j]-a[i][k]*a[k][j];
           }
       for(i=0;i<=n-1;i++)
           if(i!=k)
           {
                u=i*n+k;a[i][k]=-a[i][k]*a[k][k];
           }
    }
for(k=n-1;k>=0;k--)
{
  if(js[k]!=k)
          for(j=0;j<=n-1;j++)
```

```
            {
                u=k*n+j;v=js[k]*n+j;
                p=a[k][j];a[k][j]=a[js[k]][j];a[js[k]][j]=p;
            }
    if(is[k]!=k)
        for(i=0;i<=n-1;i++)
        {
            u=i*n+k;v=i*n+is[k];
            p=a[i][k];a[i][k]=a[i][is[k]];a[i][is[k]]=p;
        }
            }
            free(is);free(js);
            return (1);
        }
```

iii. 矩阵求逆函数的调用(C 语言)

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>

    int invGJ(double **a,int n);

    void main()
{
    int i,j;
    double **AA;

    //首先对二维指针 Naa 分配内存，采用 C 语言的方法
/*  AA=(double **)malloc(sizeof(double)*2);
    for(i=0;i<2;i++)
    {
        AA[i]=(double *)mallo(sizeof(double)*2);
    }
```