将C源程序移植到VC开发环境下的实现

吴 焱
(昆明冶金高等专科学校 计算机与信息工程系,云南 昆明 650033)

摘要:针对C程序的特点,给出将之移植到VC集成环境下的技术,对一个常用程序集实施了改写,并提供了C++数组和矩阵模板类,对C程序进行面向对象的封装.移植后程序运行正常,因此,大量经过严格检验的实用C源程序,在VC环境中又具有了新的生命力.也大大减少了软件重复开发的代价.

关键词:数值计算;封装;模板类;C源程序
中图分类号:TP311.54 **文献标识码**:A **文章编号**:1007-855X(2004)02-0072-05

Realizing the Migration of Old C Code in Visual C++ IDE

WU Yan
(Department of Computer and Information Engineering, Kunming Metallurge College, Kunming 650033, China)

Abstract: According to the characteristics of C programs, some techniques to migrate them to Visual C++ IDE as an implementation are presented and a set of numerical arithmetic programs for further engineering application is reprogrammed. The programs can run normally after the migration, so lots of practical C programs have been conferred a new life in Visual C++ IDE, and the cost paid for the repeated design of the software is lowered.
Key words: numerical arithmetic; encapsulation; template class; C language

0 引言

由于C语言长期广泛应用,现存有大量经过严格检验的实用C源程序,它们可以用来很好地解决工程应用中的实际问题.但是旧的C源程序往往有很多与现代编译器不兼容的地方,因此要进行相应的移植处理.

本文以改写清华大学出版社出版的C常用算法程序集(以下简称"程序集")为例,说明如何将旧的C源程序移植到目前普遍使用的C/C++开发环境Visual C++下.除了列举一些移植程序的方法和技巧外,本文还给出两个C++类:数组类和矩阵模板类,以例示总如何对C源程序进行面向对象的包装处理.

1 基于C语言分析和改换

Visual C++支持ANSI C,下面列举源代码影响编译、不兼容的情况和相应解决方案,并给出基于ANSI C标准的函数的基本调用例子.

1.1 改变函数定义中参数声明的形式

函数定义中参数声明时没有采用现代风格,例如全选主元高斯消去法:

```
int agaus(a,b,n)
int n;
double a[],b[];
{……;}
```

收稿日期:2003-11-14.
作者简介:吴焱(1969.12~),女,工学学士,讲师.主要研究方向:计算机教学与应用软件开发.
E-mail:wuyan69@vip.km169.net

参数声明应改为数组形式:

```
int agaus(double a[],double b[],int n)
```

或者改为指针形式:

```
int agaus(double * a,double * b,int n);
```

调用方法:

```
agaus(&a[0][0],&b[0],n);
/* a 为二维双精度型数组,b 为一维双精度型数组,n 整型变量 */
```

C/C++中用下标法和指针法都可以访问一个数组,设有数组 a,则 $a[i]$和 $*(a+i)$无条件等价.如果指针变量 p 指向数组中的一个元素,则 $p+1$ 指向同一数组的下一个元素.若 p 的初值为&$a[0]$,则 $p+i$ 和 $a+i$ 都是 $a[i]$的地址;$*(p+i)$和 $*(a+i)$就是 $p+i$ 或 $a+i$ 所指向的数组元素,即 $a[i]$;指向数组的指针变量也可以带下标,如 $p[i]$与 $*(p+i)$等价.所以,在实际使用该函数,如果遇到数组作形参,可以将数组第一个元素地址作为实参传值调用函数.

1.2 动态存储分配函数返回值类型强制转换

动态存储分配函数返回 void * 型指针变量,它指向一个抽象类型的数据,ANSI C 标准规定在将它赋值给另一个指针变量时需要进行强制类型转换,所以下面代码 Line1 要用 Line2 替换:

```
double * v;
v = malloc(n * m * sizeof(double));/* Line1 */
v = (double * )malloc(n * m * sizeof(double)); /* Line2 */
```

1.3 用函数名作为函数调用的实参

某些算法函数可能要调用一些用户自定义函数,如最佳一致逼近的里米兹方法:

```
void hremz(a,b,p,n,eps)
int n;
double a,b,eps,p[];
{ extern double hremzf();
…
}
```

原方法使程序集与应用程序的耦合程度增加,缺乏灵活性,可以改为:

```
void hremz(double a,double b,double p[],int n,double eps,double ( * hremzf)(double x))
{
…
}
```

用函数指针作参数,调用时直接将函数名作实参即可:

```
hremz(a,b,p,4,eps,hremzf);
/* 假设各参数在主程序文件已定义 */
```

1.4 将函数的输出作为字符串值返回

有的时候需要将一些函数的输出作为字符串值返回,比如:

```
printf("%c",xy[i][j]);
```

我们可以用形似

```
sprintf( buffer,"%c",xy[i][j]),
strcat( str, buffer );
```

的合并语句(其中 str 是一个足够大的字符串数组参数)代替 printf("%c",xy[i][j]).

例如:

```
char * buffer;
buffer = (char * )malloc(n * sizeof(char)); /* n 作为参数传递,例如 100 */
sprintf( buffer,"%c",xy[i][j]),
```

```
strcat( str, buffer );
/*把终端输出字符添加到 str 串尾*/
…
free(buffer)
```

如果用到了它们,调用方法以随机样本分析为例:

```
char str[1024];
str[0] = '\0';/*初始化为空串*/
irhis(x,100,x0,h,10,1,&dt[0],&g[0],&q[0],str);
```

现在 str 数组保存了终端输出文本,可以随意使用它,比如在控制台程序里输出:

```
puts(str);
```

在使用 MFC 类库时,str 可以直接赋值给一个 CString 对象的实例.

经过以上的工作,我们得到基于 ANSI C 标准的程序版本,可以在 C 和 C++开发环境下使用.

2 基于 C++语言分析和改换

由于 C 语言本身的弱点,程序集存在的缺陷主要有:异常处理机制支持较弱和程序没有对数组下标是否越界的检测.

如果编程人员对 C/C++语言很生疏,并且不熟悉该程序集,那么有可能由于编码的失误在调试过程中得到不可预知的荒谬结果.我们的解决方案是为程序集增加 C++的异常处理机制,以及用类封装技术,对数组进行面向对象的封装和使用,用 Array 模板类对象替换一维数组,用 Matrix 模板类对象替换二维数组.下面给出两个类的声明部分,它们分别实现最基本的数组和矩阵数据结构和算法.

```
template < class T = double >
class TArray
{
protected:
T* pdata;
unsigned int length;
public:
TArray();
TArray(unsigned int);
TArray(TArray const&);
virtual TArray();
void operator = (TArray&);
TArray<T>& operator + (TArray&);
TArray<T>& operator - (TArray&);
T const& operator [] (unsigned int)const;
T& operator [](unsigned int);
T const* GetData() const;
unsigned int GetLenght();
void SetLength(unsigned int,bool = true);
};
template < class T = double >
class TMatrix
{
protected:
unsigned int numberOfRows;
```

```
unsigned int numberOfColumns;
TArray < T > array;
public:
class Row
{
TMatrix < T > & matrix;
unsigned int const row;
public:
Row (TMatrix < T > &_matrix,unsigned int_row):matrix(_matrix),row(_row){}
T& operator [](unsigned int column)const
{
return matrix.Select(row,column);}
};
TMatrix();
TMatrix(unsigned int, unsigned int);
TMatrix(TMatrix < T > & mat);
virtual TMatrix();
T& Select(unsigned int, unsigned int);
Row & operator[](unsigned int);
TMatrix < T > & operator + (TMatrix < T > & mat);
TMatrix < T > & operator - (TMatrix < T > & mat);
TMatrix < T > & operator * (TMatrix < T > & mat);
bool operator = = (TMatrix < T > & mat);
TArray < T > & GetData();
unsigned int GetNumberOfRows();
unsigned int GetNumberOfColumns();
bool LoadFromArray(T [],unsigned int,unsigned int);
bool LoadFromString(char * ,char,char);
bool ResetMatrix(unsigned int, unsigned int);
bool ReverseMatrix();
void ZeroMatrix();
void RandomMatrix(int max);
};
```

举例说明关于异常机制和数组越界的检测方法的思路.TMatrix 类的 operator[]返回一个嵌套类 Row 的引用,它用来描述某一给定二维数组的一个特定行.Row 类的 operator[]则返回该行一个特定位置的 T 类型值.最终实现还是通过 Matrix < T > ::Select()函数,该函数具体代码如下:

```
template < class T >
T& TMatrix < T > ::Select(unsigned int i, unsigned int j)
{
char ch[50];
if (i > = numberOfRows)
sprintf(ch," Error - - Invalid row: %d",i), throw (ch);
if (j > = numberOfColumns)
sprintf(ch," Error - - Invalid colum: %d",j), throw (ch);
return array[i * numberOfColumns + j];
```

```
}
应用程序实例:
unsigned int i,j;
unsigned int m = 2,n = 3;
TMatrix < > mat(m,n);//双精度型矩阵
{
for(i = 0; i < m; i + + )
{
for(j = 0; j < n + 1; j + + )// Line3
cout < < mat[i][j] < < " \ t";
cout < < endl;
}
}
catch(char * str) //捕获异常
{cout < < str < < endl;}
```

终端输出如下(注:类实例 mat 没有初始化):

-6.27744e+066 -6.27744e+066 -6.27744e+066 Error - - Invalid colum: 3

只输出一行,根据出错提示,把 Line3 改为:"for(j = 0; j < n; j + +)",重新编译运行,输出 2 行 3 列的正确结果:

-6.27744e+066 -6.27744e+066 -6.27744e+066

-6.27744e+066 -6.27744e+066 -6.27744e+066

由于我们对 operator[]进行了重载,所以数组模板类(矩阵模板类)完全兼容 C/C + + 一维数组(二维数组)的存取操作,因此旧程序中数组变量直接可以用类实例变量替代.

上边给出的数组和矩阵模板类简洁高效,易于使用,并且也有较好的灵活性.但是,标准 C + + 中提供了为数值计算优化过的 valarray 模板类取代数组操作,请参阅文献[3],在 Numerics 这一章中,探讨了 valarray,slice 等技术,并给出一个在此基础上 Matrix 类的实现.建议熟练 C + + 的程序员掌握这些更为权威、高级的技术.

3 结 论

新的程序与原程序相比较有如下优点:

1) 遵从 ISO C/C + + 标准,因此具有良好的可移植性.可以在大多数流行的 C + + 开发环境下使用;

2) 利用一些技巧,改进了原程序不利于扩展和缺少灵活性的缺点;

3) 去除了原程序中几个影响效率的 Bug;

4) 增加异常处理机制和数组越界检测,增强可调试性和健壮性;

5) 数组和矩阵操作得到了强有力的支持.

经过实际应用测试,新的程序集可以满足一般工程应用的数值计算需要,并且能够在原来的基础上,方便地进行必要的改进和扩充.因此大量经过严格检验的实用 C 源程序,在 VC 环境中又具有了新的生命力,也大大减少了软件重复开发的代价.

参考文献:

[1] 徐士良. C 常用算法程序集[M].北京:清华大学出版社, 1996.2 ~ 225.

[2] 谭浩强. C 程序设计[M]. 北京:清华大学出版社,1998.38 ~ 65.

[3] Bjarne Stroustrup. The C + + Programming Language: Special Edition[M]. 北京:机械工业出版社,2002.640 ~ 671.

[4] Stanley B.Lippman. C + + Primer[M].北京:中国电力出版社,2002.12 ~ 65.

[5] Andrew Koenig Barbara.C 陷阱与缺陷[M].北京:人民邮电出版社,2002.1 ~ 172.