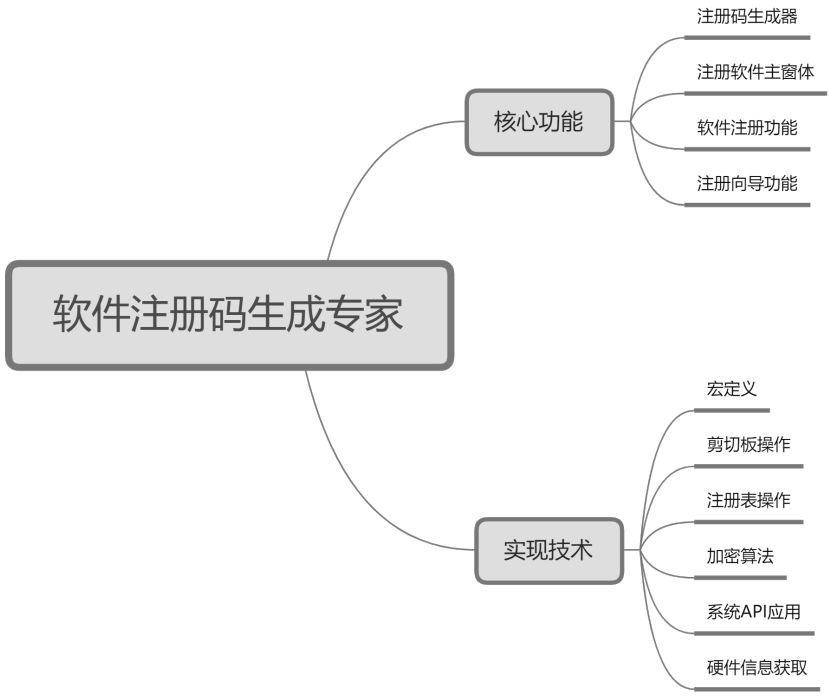
第 2 章

软件注册码生成专家

——宏技术 + 剪贴板操作 + 注册表操作+加密算法+系统 API 应用+硬件信息获取

在当今时代，计算机已成为不可或缺的工具，无论是在工作还是生活中，其应用都日益广泛。在众多使用计算机的领域中，安装应用软件已成为常态。随着软件用户群体的不断扩大，盗版软件问题日益严重，成为软件开发商的一大难题。为了防止软件被盗用，软件在安装后通常要求用户进行注册。未注册的用户无法使用软件，从而有效防止软件被盗用。本章采用 C++语言进行开发，其中：注册表操作技术用于生成注册码并限制试用次数；加密算法技术用于对注册码进行加密；系统 API 应用和硬件信息获取技术用于在不同计算机上获取唯一序列号。本章还结合宏技术、剪贴板操作技术等关键技术，开发一个名为“软件注册码生成专家”的项目，旨在防止软件被盗用。

本项目的核心功能及实现技术如下：



2.1 开 发 背 景

在当今计算机时代，软件已经成为工作和生活中不可或缺的一部分。随着用户群体的不断扩大，盗版软件问题日益严重，成为软件开发商的头号难题。为了保护软件版权，开发者通常会要求用户进行注册并获取

注册码后才能正常使用软件，这一举措有效防止了软件的盗用和非法传播。在这样的背景下，开发注册码生成器成为迫切需求，旨在确保软件合法使用，维护软件行业的良性发展。这背后涉及加密技术、用户体验和防盗版策略的综合考量，是软件开发领域中的重要挑战之一。

本项目实现目标如下：

- ☑ 获取 CPU 序列号。
- ☑ 获取网卡地址。
- ☑ 生成注册码。
- ☑ 获取磁盘序列号。
- ☑ 网站后台管理。

2.2 系 统 设 计

2.2.1 开发环境

本项目的开发及运行环境如下：

- ☑ 操作系统：推荐 Windows 10、Windows 11 或更高版本。
- ☑ 开发工具：Visual Studio 2022。
- ☑ 开发语言：C++/Win32API。

2.2.2 业务流程

在启动项目后，首先使用注册码生成器生成一个注册码，然后运行软件注册功能。用户如果已有注册码，则需要输入用户名和注册码进行软件注册，注册成功之后，系统会提示“注册成功”；用户如果没有注册码，可以选择试用（试用次数为 30 次），系统同样会提示“注册成功”。

本项目的业务流程如图 2.1 所示。

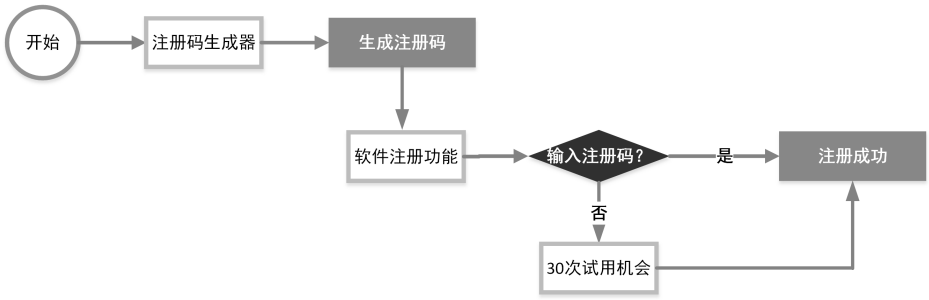


图 2.1 软件注册码生成专家业务流程

2.2.3 功能结构

本项目的功能结构已经在章首页中给出。本项目实现的具体功能如下：

- ☑ 注册码生成器：生成注册码。
- ☑ 注册软件主窗口：判断模块是否经过注册，如果没有注册，则在主窗体显示前显示注册模块。
- ☑ 软件注册功能：在运行软件注册模块时，如果模块没有经过注册，则会显示注册模块，在注册模块中，用户可以选择使用已有的注册码进行注册，也可以选择试用软件。

☑ 注册向导：用户输入用户名和注册码，单击“注册”按钮后，程序会自动检测用户名和注册码是否正确。如果输入正确，程序将允许用户进入主窗体；否则，会弹出提示。

2.3 技 术 准 备

2.3.1 技术概览

☑ 宏技术：宏可以用于定义常量或简单的内联函数，以提高代码的可读性和可维护性。

例如：定义一个宏，代码如下：

```
#define SQUARE(x) ((x) * (x))
```

☑ 剪贴板操作：需要使用操作系统提供的 API 或者通过第三方库来实现。在 Windows 平台下，可以使用 Windows API 来进行剪贴板操作。

☑ 注册表操作：使用注册表来存储和检索应用程序的设置。MFC 提供了一些类和函数来简化注册表操作，这些功能主要是通过 CWinApp 类的成员函数和 CRegKey 类来实现的。

☑ 加密算法：加密算法用于将数据转换为不可读或难以理解的形式，以保护数据的安全性。在 C++ 中，可以使用各种加密算法来加密和解密数据。

☑ 系统 API 应用：用于与操作系统进行交互和控制系统资源。在 C++中，系统 API 可用于执行多种任务，包括文件操作、进程管理、网络通信等。

☑ 硬件信息获取：获取硬件信息通常涉及与操作系统进行交互以查询系统资源和设备信息。在 Windows 平台下，可以使用系统 API 或 WMI 来获取硬件信息。本项目采用 API 获取硬件信息，获取的信息将会在 2.3.2 节~2.3.4 节中进行介绍。

《Visual C++从入门到精通（第 5 版）》详细地讲解了宏技术、剪贴板操作、注册表操作、加密算法、系统 API 应用等知识，对这些知识不太熟悉的读者，可以参考该书对应的内容。

2.3.2 获取 CPU 序列号

在软件注册模块中，注册码是通过硬件信息生成的，其中包括 CPU 序列号。实现代码如下：

```
CString CCreateRegDlg::GetCPUNum()
{
    //获取 CPU 序列号
    unsigned long s1, s2;

    CString CPUID1, CPUID2;
    __asm{
        mov eax, 01h        //将 eax 寄存器的值设置为 1，用于指定 cpuid 指令的功能码
        xor edx, edx        //将 edx 寄存器的值清零
        cpuid               //执行 cpuid 指令，获取 CPU 信息
        mov s1, edx         //将 edx 寄存器的值移动到 s1 变量中
        mov s2, eax         //将 eax 寄存器的值移动到 s2 变量中
    }

    CPUID1.Format("%08X%08X", s1, s2);
    __asm{
        mov eax, 03h        //将 eax 寄存器的值设置为 3，用于指定另一个 cpuid 指令的功能码
        xor ecx, ecx        //将 ecx 寄存器的值清零
        xor edx, edx        //将 edx 寄存器的值清零
        cpuid               //执行 cpuid 指令，获取更多 CPU 信息
        mov s1, edx         //将 edx 寄存器的值移动到 s1 变量中
        mov s2, ecx         //将 ecx 寄存器的值移动到 s2 变量中
    }
```

```
    CPUID2.Format("%08X%08X", s1, s2);

    //拼接结果
    CString CpuID = CPUID1 + CPUID2;
    return CpuID.Mid(5, 3);     //从拼接的结果中取出一段,作为返回结果
}
```

说明

通过使用 cpuid 指令，我们可以获取 CPU 序列号以及处理器的详细信息。cpuid 指令是自 Intel 486 处理器以来就被加入支持的功能。

2.3.3 获得磁盘序列号

在软件注册模块中，注册码是通过硬件信息生成的，其中包括磁盘序列号。通过函数 GetVolumeInformation，我们可以获取磁盘驱动器的磁盘列号。实现代码如下：

```
BOOL GetVolumeInformation(
      LPCTSTR lpRootPathName,
      LPTSTR lpVolumeNameBuffer,
      DWORD nVolumeNameSize,
      LPDWORD lpVolumeSerialNumber,
      LPDWORD lpMaximumComponentLength,
      LPDWORD lpFileSystemFlags,
      LPTSTR lpFileSystemNameBuffer,
      DWORD nFileSystemNameSize
);
```

参数说明如下：

- ☑ lpRootPathName：指定要获取信息的磁盘的根路径。
- ☑ lpVolumeNameBuffer：用于存储磁盘名的字符串缓冲区。
- ☑ nVolumeNameSize：lpVolumeNameBuffer 缓冲区的大小。
- ☑ lpVolumeSerialNumber：用于存储磁盘序列号的变量。
- ☑ lpMaximumComponentLength：用于存储文件名每一部分的长度。
- ☑ lpFileSystemFlags：用于存储一个或多个二进制位标志的变量。
- ☑ lpFileSystemNameBuffer：指定一个缓冲区，用于存储文件系统的名称。
- ☑ nFileSystemNameSize：lpFileSystemNameBuffer 缓冲区的大小。

获得磁盘序列号的实现代码如下：

```
CString CCreateRegDlg::GetDiskNum()
{
    DWORD ser;
    char namebuf[128];
    char filebuf[128];
    //获取 C 盘的序列号
    ::GetVolumeInformation("c:\\",          //指定要获取信息的磁盘的根路径
                           namebuf,         //用于存储磁盘名的字符串缓冲区
                           128,             //namebuf 缓冲区的大小
                           &ser,            //用于存储磁盘序列号的变量
                           0,               //用于存储文件名每一部分的长度
                           0,               //标志位
                           filebuf,         //用于存储文件系统的名称
                           128              //上面缓冲区的大小
                          );
    CString DiskID;
    //格式化成字符串
    DiskID.Format("%08X", ser);
    //返回第 3 个开始的 3 个字符
```

```
    return DiskID.Mid(3, 3);
}
```

2.3.4 获得网卡地址

在软件注册模块中，注册码是通过硬件信息生成的，其中包括网卡地址。为了获得网卡的地址，程序需要调用 Netbios 函数，并在该函数调用时设置 NCBASTAT 命令。Netbios 函数的原型定义如下：

```
UCHAR
APIENTRY
Netbios(
    PNCB pncb
);
```

实现代码如下：

```
CString CCreateRegDlg::GetMacAddress()
{
    NCB nInfo;
    //内容清零
    memset(&nInfo, 0, sizeof(NCB));
    //设置命令
    nInfo.ncb_command    = NCBRESET;
    nInfo.ncb_lana_num = 0;
    //执行
    Netbios(&nInfo);

    ADAPTER_INFO AdaINfo;
    //初始化 NetBIOS
    memset(&nInfo, 0, sizeof(NCB));
    nInfo.ncb_command    = NCBASTAT;
    nInfo.ncb_lana_num = 0;
    nInfo.ncb_buffer     = (unsigned char *)&AdaINfo;
    nInfo.ncb_length     = sizeof(ADAPTER_INFO);
    strncpy((char *)nInfo.ncb_callname, "*", NCBNAMSZ);
    Netbios(&nInfo);

    //格式化为字符串
    CString MacAddr;
    MacAddr.Format("%02X%02X%02X%02X%02X%02X",
                   AdaINfo.nStatus.adapter_address[0],
                   AdaINfo.nStatus.adapter_address[1],
                   AdaINfo.nStatus.adapter_address[2],
                   AdaINfo.nStatus.adapter_address[3],
                   AdaINfo.nStatus.adapter_address[4],
                   AdaINfo.nStatus.adapter_address[5]
                  );
    //返回一段字符串
    return MacAddr.Mid(4, 4);
}
```

2.3.5 生成注册码

在获得了各个硬件的数据之后，我们需要利用这些数据来生成注册码。首先，我们声明一个密钥数组。实现代码如下：

```
    //定义一个密钥数组
    CString code[16] = {"ah", "tm", "ib", "nw", "rt", "vx", "zc", "gf",
                        "pn", "xq", "fc", "oj", "wm", "eq", "np", "qw"
                       };
```

然后，我们将硬件数据转换为十六进制数据，并据此从密钥数组中读取相应的密钥。实现代码如下：

```
CString reg, stred;
int num;
stred = GetCPUNum() + GetDiskNum() + GetMacAddress();
stred.MakeLower();
//根据十六进制数字从密钥数组中选择对应的字符串
for(int i = 0; i < 10; i++) {
    char p = stred.GetAt(i);
    if(p >= 'a' && p <= 'f') {
        num = p - 'a' + 10;
    }
    else {
        num = p - '0';
    }
    CString tmp = code[num];
    reg += tmp;
}

//结果转化成大写
reg.MakeUpper();
```

2.3.6 根据注册表中数据限制试用次数

软件注册模块提供了注册和试用两种运行模式供用户选择。如果用户选择试用，系统将首先判断用户是否已经试用过，若用户未曾试用，系统将自动设置允许用户试用 30 次；如果用户已经试用过，系统将读取注册表中的剩余试用次数，并向用户显示他们可用的剩余试用次数。实现代码如下：

```
HKEY key;
char data[4];
DWORD size = 4;
DWORD type = REG_DWORD;

//REG_DWORD = REG_SZ;
CString skey = "Software\\mingrisoft";
LSTATUS iret = RegOpenKeyEx(HKEY_CURRENT_USER, skey,
                            REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, &key);
if(iret == 0) {
    CString value;
    //读取试用次数
    iret = RegQueryValueEx(key, "tryout", NULL, &type, (BYTE *)data, &size);
    if(iret == 0) {
        if(data != 0) {
            CString strTime;
            //在界面上显示试用次数
            strTime.Format("你还可以使用%s 次", data);
            GetDlgItem(IDC_STATICTIME)->SetWindowText(strTime);
        }
        else {
            //界面上的控件设置为不可用
            GetDlgItem(IDC_RADIO2)->EnableWindow(FALSE);
            //提示不可以试用软件
            GetDlgItem(IDC_STATICTIME)->SetWindowText("你已经不可以再试用本软件了！");
        }
    }
    else {
        //设置试用次数为 30
        RegSetValueEx(key, "tryout", NULL, REG_DWORD, (BYTE *)"30", 2);
        OnCancel();
    }
    RegCloseKey(key);
}
```

选择试用软件以后，系统会将减少后的试用次数写入注册表中。实现代码如下：

```
    HKEY key;
    CString skey = "Software\\mingrisoft";
    long iret = RegOpenKeyEx(HKEY_CURRENT_USER,
```

```
                                    skey,
                                    REG_OPTION_NON_VOLATILE,
                                    KEY_ALL_ACCESS,
                                    &key);
    if(iret == 0) {
        //从界面上获得试用次数相关文字
        CString str;
        GetDlgItem(IDC_STATICTIME)->GetWindowText(str);
        CString num;
        //提取试用次数并转换为整型数字
        int run = _ttoi(str.Mid(12, str.GetLength() - 14));
        num.Format("%d", run - 1);
        //写入注册表
        RegSetValueEx(key, "tryout", 0, REG_SZ, (BYTE *)num.GetBuffer(0)
                      , num.GetLength()*sizeof(TCHAR));
        //设置全局标志位
        Flag = TRUE;
        CDialog::OnOK();
        RegCloseKey(key);
    }
```

2.3.7 注册快捷键

要为程序设置快捷键，可以使用 RegisterHotKey 函数来根据用户设置的热键组合进行注册。RegisterHotKey 函数的语法格式如下：

```
BOOL RegisterHotKey( HWND hWnd, int id,UINT fsModifiers, UINT vk );
```

参数说明：

- ☑ hWnd：注册快捷键的窗体句柄。
- ☑ id：用户的自定义消息 ID 值。
- ☑ fsModifiers：设置组合键，取值如下：
  - ➢ MOD_ALT：Alt 键。
  - ➢ MOD_CONTROL：Ctrl 键。
  - ➢ lMOD_SHIFT：Shift 键。
  - ➢ MOD_WIN：Win 键。
- ☑ vk：快捷键的虚拟键代码。

当程序运行结束时，需要注销已经注册的快捷键，这可以通过 UnregisterHotKey 函数来实现。UnregisterHotKey 函数的语法格式如下：

```
BOOL UnregisterHotKey( HWND hWnd, int id );
```

参数说明：

- ☑ hWnd：注册快捷键的窗体句柄。
- ☑ id：用户的自定义消息 ID 值。

注册快捷键的步骤如下。

（1）声明自定义消息。实现代码如下：

```
#define HOTKEY_PASTE            11111
```

（2）添加快捷键消息的函数声明。实现代码如下：

```
afx_msg void OnHotKey(WPARAM wParam,LPARAM lParam);
```

（3）添加消息映射。实现代码如下：

```
ON_MESSAGE(WM_HOTKEY,OnHotKey)
```

（4）添加消息响应函数的实现代码。实现代码如下：

```
void CRegisterNumDlg::OnHotKey(WPARAM wParam,LPARAM lParam)
{
    if(HOTKEY_PASTE == (int)wParam)                    //快捷键消息
    {
        PasteReg();                                    //实现的功能
    }
}
```

2.3.8 一次性粘贴注册码

在使用软件注册模块时，用户可以一次性粘贴注册码，这一功能是通过剪贴板来实现的。在程序中添加两个消息映射宏 ON_WM_CHANGECBCHAIN 和 ON_WM_DRAWCLIPBOARD，用于实时监测剪贴板中的内容。其中，ON_WM_DRAWCLIPBOARD 宏负责显示剪贴板中的内容，而 ON_WM_CHANGECBCHAIN 宏则用于控制是否继续对剪贴板进行监视。随后，使用 OpenClipboard 函数来打开剪贴板。剪贴板打开后，可以通过 GetClipboardData 函数来获取剪贴板中的内容。操作完成后，利用 CloseClipboard 函数关闭剪贴板。

（1）添加消息映射。实现代码如下：

```
ON_WM_CHANGECBCHAIN()
ON_WM_DRAWCLIPBOARD()
```

（2）添加函数声明。实现代码如下：

```
afx_msg void OnDrawClipboard();
afx_msg void OnChangeEdit1();
```

（3）添加 OnChangeCbChain 函数的实现代码，用于停止对剪贴板进行监视。实现代码如下：

```
void CRegisterNumDlg::OnChangeCbChain(HWND hWndRemove, HWND hWndAfter)
{
    if( hwnd==hWndRemove )
        hwnd=hWndAfter;
    ::SendMessage(hwnd,WM_CHANGECBCHAIN,(WPARAM)hWndRemove,(LPARAM)hWndAfter);
}
```

（4）添加 OnDrawClipboard 函数的实现代码，用于获取剪贴板中的数据。实现代码如下：

```
void CRegisterNumDlg::OnDrawClipboard()
{
    CString edit,str;
    OpenClipboard();                                   //打开剪贴板
    if(IsClipboardFormatAvailable(CF_TEXT))            //判断剪贴板中是否有文本格式数据
    {
        HANDLE hmem = ::GetClipboardData(CF_TEXT);     //获得剪贴板中数据
        char*data     = (char*)GlobalLock(hmem);
        str = data;
        if(m_Once)
            GetDlgItem(IDC_EDIT1)->SetWindowText(str); //粘贴用户名
        else
        {
            int index = str.Find("-");                 //查找注册码中的"-"
            if(index == -1 || str.GetLength() != 23)   //判断注册码格式是否正确
            {
                MessageBox(_T("注册码格式不正确！"));
                return;
            }
            else
            {
                int i = 0;                             //用于获取编辑框位置
                while(index != -1)
                {
                    edit = str.Left(5);                //获得注册码
                    GetDlgItem(IDC_EDIT2+i++)->SetWindowText(edit); //在对应编辑框中显示
                    str = str.Mid(index + 1);
                    index = str.Find("-");
                }
```

```
                edit = str;
                GetDlgItem(IDC_EDIT2+i)->SetWindowText(edit);          //显示最后一组注册码
            }
        }
    }
    CloseClipboard();                                                  //关闭剪贴板
    ::SendMessage(hwnd,WM_DRAWCLIPBOARD,0,0);                          //重新发送剪贴板消息
}
```

（5）添加 PasteReg 方法，用于调用 OnChangeCbChain 函数和 OnDrawClipboard 函数。实现代码如下：

```
void CRegisterNumDlg::PasteReg()
{
    hwnd = SetClipboardViewer();                                       //设置剪贴板查看器
    Sleep(100);                                                        //延时
    ChangeClipboardChain(hwnd);                                        //停止检测剪贴板
}
```

2.4 注册码生成器模块

2.4.1 注册码生成器模块概述

注册码生成器模块通过计算机的 CPU 序列号、C 盘序列号和网卡地址来生成一组注册码，用户在使用软件注册模块时可以通过这组注册码和对应的用户名来填写注册信息，从而完成软件的注册过程，如图 2.2 所示。

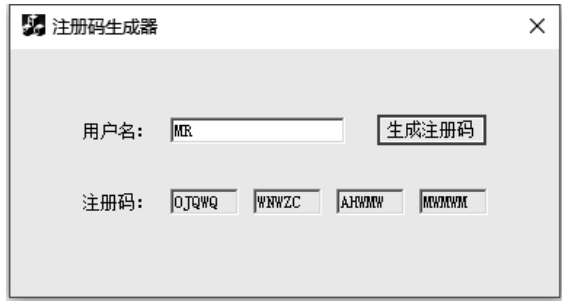


图 2.2 注册码生成器模块

2.4.2 界面设计

注册码生成器模块界面设计过程如下：

（1）创建一个基于对话框的应用程序。

（2）向对话框中添加控件，包括两个静态文本控件、五个编辑框控件和一个按钮控件，控件的属性设置如表 2.1 所示。

表 2.1 注册码生成器模块的控件属性设置表

控件 ID	控件属性	关联变量
IDC_STATIC	Caption：用户名、Simple	无
IDC_STATIC	Caption：密码、Simple	无
IDOK	Bitmap、Flat	无
IDC_EDIT1	选中 Uppercase	无
IDC_EDIT2	选中 read-only	无
IDC_EDIT3	选中 read-only	无
IDC_EDIT4	选中 read-only	无
IDC_EDIT5	选中 read-only	无

2.4.3 获取序列号

注册码生成器模块获取序列号过程如下。

（1）添加 GetCPUNum 方法，用于获取 CPU 序列号中从第 5 个字符起的 3 个字符。实现代码如下：

```
CString CCreateRegDlg::GetCPUNum()
{
    //获取 CPU 序列号
    unsigned long s1, s2;

    CString CPUID1, CPUID2;
    __asm{
        mov eax, 01h        //将 eax 寄存器的值设置为 1，用于指定 cpuid 指令的功能码
        xor edx, edx        //将 edx 寄存器的值清零
        cpuid               //执行 cpuid 指令，获取 CPU 信息
        mov s1, edx         //将 edx 寄存器的值移动到 s1 变量中
        mov s2, eax         //将 eax 寄存器的值移动到 s2 变量中
    }

    CPUID1.Format("%08X%08X", s1, s2);
    __asm{
        mov eax, 03h        //将 eax 寄存器的值设置为 3，用于指定另一个 cpuid 指令的功能码
        xor ecx, ecx        //将 ecx 寄存器的值清零
        xor edx, edx        //将 edx 寄存器的值清零
        cpuid               //执行 cpuid 指令，获取更多 CPU 信息
        mov s1, edx         //将 edx 寄存器的值移动到 s1 变量中
        mov s2, ecx         //将 ecx 寄存器的值移动到 s2 变量中
    }
    CPUID2.Format("%08X%08X", s1, s2);

    //拼接结果
    CString CpuID = CPUID1 + CPUID2;
    return CpuID.Mid(5, 3);     //从拼接的结果中取出一段,作为返回结果
}
```

（2）添加 GetDiskNum 方法，用于获取 C 盘序列号中从第 3 个字符起的 3 个字符。实现代码如下：

```
CString CCreateRegDlg::GetDiskNum()
{
    DWORD ser;
    char namebuf[128];
    char filebuf[128];
    //获取 C 盘的序列号
    ::GetVolumeInformation("c:\\",          //欲获取信息的磁盘的根路径
                           namebuf,         //用于存储磁盘名的缓冲区
                           128,             //上面缓冲区的大小
                           &ser,            //用于存储磁盘序列号的变量
                           0,               //用于存储文件名每一部分的长度
                           0,               //标志位
                           filebuf,         //用于存储文件系统的名称
                           128              //上面缓冲区的大小
                          );
    CString DiskID;
    //格式化成字符串
    DiskID.Format("%08X", ser);
    //返回第 3 个开始的 3 个字符
    return DiskID.Mid(3, 3);
```

（3）在工程中引入头文件 nb30.h 和动态链接库 netapi32.lib。实现代码如下：

```
#include "nb30.h"
#pragma comment (lib,"netapi32.lib")
```

（4）声明一个结构 ADAPTER_INFO，用来存储网卡信息。实现代码如下：

```
struct ADAPTER_INFO {
    ADAPTER_STATUS nStatus;
    NAME_BUFFER    nBuffer;
}
```

（5）添加 GetMacAddress 方法，用于获取网卡地址中从第 4 个字符起的 4 个字符。实现代码如下：

```
CString CCreateRegDlg::GetMacAddress()
{
    NCB nInfo;
    //内容清零
    memset(&nInfo, 0, sizeof(NCB));
    //设置命令
```

```
    nInfo.ncb_command    = NCBRESET;
    nInfo.ncb_lana_num = 0;
    //执行
    Netbios(&nInfo);

    ADAPTER_INFO AdaINfo;
    //初始化 NetBIOS
    memset(&nInfo, 0, sizeof(NCB));
    nInfo.ncb_command    = NCBASTAT;
    nInfo.ncb_lana_num = 0;
    nInfo.ncb_buffer     = (unsigned char *)&AdaINfo;
    nInfo.ncb_length     = sizeof(ADAPTER_INFO);
    strncpy((char *)nInfo.ncb_callname, "*", NCBNAMSZ);
    Netbios(&nInfo);

    //格式化成字符串
    CString MacAddr;
    MacAddr.Format("%02X%02X%02X%02X%02X%02X",
                   AdaINfo.nStatus.adapter_address[0],
                   AdaINfo.nStatus.adapter_address[1],
                   AdaINfo.nStatus.adapter_address[2],
                   AdaINfo.nStatus.adapter_address[3],
                   AdaINfo.nStatus.adapter_address[4],
                   AdaINfo.nStatus.adapter_address[5]
                  );
    //返回一段字符串
    return MacAddr.Mid(4, 4);
}
```

2.4.4 实现“生成注册码”按钮功能

处理“生成注册码”按钮的单击事件，根据 CPU 序列号、磁盘序列号和网卡地址生成注册码，并将用户名和注册码写入程序根目录下的 sn.txt 文件中。实现代码如下：

```
void CCreateRegDlg::OnOK()
{
    //TODO: Add extra validation here
    CString name;
    GetDlgItem(IDC_EDIT1)->GetWindowText(name);
    if(name.IsEmpty()) {
        MessageBox("用户名不能为空！");
        return;
    }
    //定义一个密钥数组
    CString code[16] = {"ah", "tm", "ib", "nw", "rt", "vx", "zc", "gf",
                        "pn", "xq", "fc", "oj", "wm", "eq", "np", "qw"
                       };
    CString reg, stred;
    int num;
    stred = GetCPUNum() + GetDiskNum() + GetMacAddress();
    stred.MakeLower();
    //根据十六进制数字从密钥数组中选择字符串
    for(int i = 0; i < 10; i++) {
        char p = stred.GetAt(i);
        if(p >= 'a' && p <= 'f') {
            num = p - 'a' + 10;
        }
        else {
            num = p - '0';
        }
        CString tmp = code[num];
        reg += tmp;
    }

    //将结果转化为大写
    reg.MakeUpper();

    //设置界面上编辑框的内容
    GetDlgItem(IDC_EDIT2)->SetWindowText(reg.Mid(0, 5));
```

```
    GetDlgItem(IDC_EDIT3)->SetWindowText(reg.Mid(5, 5));
    GetDlgItem(IDC_EDIT4)->SetWindowText(reg.Mid(10, 5));
    GetDlgItem(IDC_EDIT5)->SetWindowText(reg.Mid(15, 5));

    //把结果写入注册表中
    HKEY key;
    CString skey = "Software\\mingrisoft"; //如果没有子项，就新建子项
    RegOpenKey(HKEY_CURRENT_USER, skey, &key);
    CString value = name + "-" + reg;
    int iret = RegSetValueEx(key, "regnum", 0, REG_SZ, (BYTE *)value.GetBuffer(0),
                                    value.GetLength());

    //只能写入 REG_SZ 型数据
    if(iret == 0) {
        MessageBox("创建成功", "提示", MB_OK);
    }
    RegSetValueEx(key, "isreg", 0, REG_SZ, (BYTE *)"0", 1);

    //将用户名和注册码写入 sn.txt 文件中
    CFile file;
    char path[256];
    ::GetCurrentDirectory(256, path);
    CString filename = path;
    filename += "\\sn.txt";
    file.Open(filename, CFile::modeCreate | CFile::modeWrite);        //打开文件
    CString text = name + "\r\n" + reg.Mid(0, 5) + "-" + reg.Mid(5, 5) +
                          "-" + reg.Mid(10, 5) + "-" + reg.Mid(15, 5);
    //写入
    file.Write(text, text.GetLength());
    //关闭文件
    file.Close();
}
```

2.5 注册软件主窗体模块

2.5.1 注册软件主窗体模块概述

在软件启动过程中，首先判断软件注册模块是否已完成注册。若检测到该模块尚未注册，则在主窗体显示之前，会弹出注册模块界面供用户进行注册。软件注册成功后的主窗体模块如图 2.3 所示。

图 2.3 软件注册成功主窗体模块

2.5.2 界面设计

软件注册模块的主窗体界面设计过程如下：

（1）创建一个基于对话框的应用程序。

（2）向对话框中添加一个图片控件，然后打开该图片控件的属性窗口，将 Type 属性设置为 Bitmap，并将 Image 属性设置为 IDB_BITMAPBK。

说明

通常，与 IDB_BITMAPBK 对应的位图文件被存储在应用程序的资源文件中，这些资源文件可以是.bmp 格式的图像文件。开发者在设计界面时会使用这些位图文件来展示背景图案、图标、按钮图片等。

2.5.3 实现注册软件主窗体功能

在软件注册模块的主窗口的 OnInitDialog 方法中，首先读取注册表信息以判断该模块是否已经注册。如果判断结果为已注册，则直接运行软件；如果判断结果为未注册，则在显示主窗体前显示注册模块。实现代码如下：

```
BOOL CCreateRegDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    //IDM_ABOUTBOX 必须在系统命令范围内
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu *pSysMenu = GetSystemMenu(FALSE);
    if(pSysMenu != NULL) {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if(!strAboutMenu.IsEmpty()) {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    //为这个对话框设置图标。框架会自动完成这个操作
    SetIcon(m_hIcon, TRUE);            //设置大图标
    SetIcon(m_hIcon, FALSE);           //设置小图标

    HKEY key;
    LPCTSTR skey = "Software\\mingrisoft";
    long iret      = RegCreateKey(HKEY_CURRENT_USER, skey, &key);
    return TRUE;   //返回 TRUE 以表示成功处理了消息
}
```

2.6 软件注册功能模块

2.6.1 软件注册功能模块概述

在运行软件注册模块时，如果模块没有经过注册，则会显示注册模块。在注册模块中，用户可以选择使用已有的注册码进行注册，也可以选择试用软件。软件注册模块如图 2.4 所示。

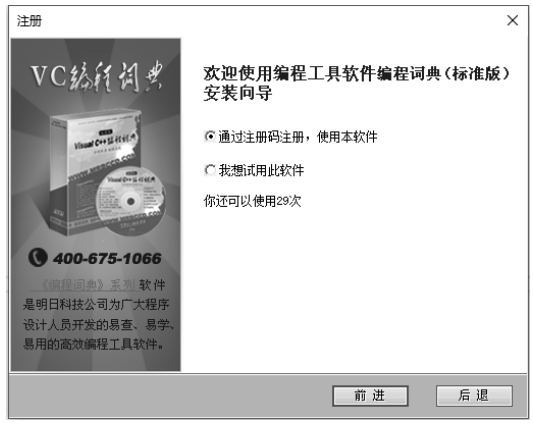


图 2.4 软件注册模块

2.6.2 界面设计

软件注册模块的注册界面设计过程如下：

（1）新建一个对话框资源。

（2）向对话框中添加控件，包括两个单选按钮控件、三个静态文本控件和两个按钮控件。这些控件的属性设置如表 2.2 所示。

表 2.2 软件注册模块的控件属性设置

控件 ID	控件属性	关联变量
IDC_STATICTIME	选中 Simple	无
IDC_RADIO1	选中 Group	Int m_Radio
IDC_RADIO2	无	
IDOK	Bitmap、Flat	CButton m_OK
IDCANCEL	Bitmap、Flat	CButton m_Cancel

2.6.3 读取试用次数

在注册模块的 OnInitDialog 方法中，读取注册表中的试用次数信息，并将用户剩余的试用次数显示出来。如果是第一次运行该模块，则设置允许用户试用 30 次。实现代码如下：

```
BOOL CSelectDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    //读取注册表
    HKEY key;
    char data[4];
    DWORD size = 4;
    DWORD type = REG_SZ;
    CString skey = "Software\\mingrisoft";
    LSTATUS iret = RegOpenKeyEx(HKEY_CURRENT_USER, skey,
                                REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, &key);
    if(iret == ERROR_SUCCESS) {
        CString value;
        //读取试用次数
        iret = RegQueryValueEx(key, "tryout", 0, &type, (BYTE *)data, &size);
        if(iret == ERROR_SUCCESS) {
            if(data[0] != 0) {
                CString strTime;
                //在界面上显示试用次数
                strTime.Format("你还可以使用%s 次", data);
                GetDlgItem(IDC_STATICTIME)->SetWindowText(strTime);
            }
            else {
                //将界面上的控件设置为不可用
                GetDlgItem(IDC_RADIO2)->EnableWindow(FALSE);
                //提示不可以试用软件
                GetDlgItem(IDC_STATICTIME)->SetWindowText("你已经不可以再试用本软件了！");
            }
        }
        else {
            //设置试用次数为 30
            RegSetValueEx(key, "tryout", 0, REG_SZ, (BYTE *)"30", 2);
            OnCancel();
        }
    }   RegCloseKey(key);
    //设置 OK 按钮的图片
    m_OK.SetBitmap(LoadBitmap(AfxGetInstanceHandle(),
                              MAKEINTRESOURCE(IDB_BITMAPOK)));
    //设置 Cancel 按钮的图片
    m_Cancel.SetBitmap(LoadBitmap(AfxGetInstanceHandle(),
                                  MAKEINTRESOURCE(IDB_BACKOFF)));
    m_Radio = 0;
    UpdateData(FALSE);
    return TRUE;  //返回 TRUE 以表示成功处理了此消息
}
```

2.6.4 实现“前进”按钮功能

处理“前进”按钮的单击事件，用户如果选择的是试用软件，则直接进入主窗体；用户如果选择的是注

册软件，则进入注册向导窗体进行注册。实现代码如下：

```
void CSelectDlg::OnOK()
{
    //TODO: Add extra validation here
    UpdateData(TRUE);
    //选中了“通过注册码注册，使用本软件”单选按钮
    if(m_Radio == 0) {
        CDialog::OnOK();
        CRegisterNumDlg dlg;
        dlg.DoModal();
    }
    //选中了“我想试用此软件”单选按钮
    else if(m_Radio == 1) {
        //打开注册表中关于试用次数的键值,准备写入
        HKEY key;
        CString skey = "Software\\mingrisoft";
        long iret = RegOpenKeyEx(HKEY_CURRENT_USER,
                                 skey,
                                 REG_OPTION_NON_VOLATILE,
                                 KEY_ALL_ACCESS,
                                 &key);
        if(iret == 0) {
            //从界面上获得试用次数相关文字
            CString str;
            GetDlgItem(IDC_STATICTIME)->GetWindowText(str);
            CString num;
            //将试用次数转换为整型数字
            int run = atoi(str.Mid(12, str.GetLength() - 14));
            num.Format("%d", run - 1);
            //写入注册表
            RegSetValueEx(key, "tryout", 0, REG_SZ, (BYTE *)num.GetBuffer(0)
                          , num.GetLength());
            //设置全局标志位
            Flag = TRUE;
            CDialog::OnOK();
        }
    }
}
```

2.7 注册向导窗体模块

2.7.1 注册向导窗体模块概述

在软件注册模块的注册向导窗体模块中，用户需要输入用户名和注册码。单击“注册”按钮后，程序将自动检测用户名和注册码是否正确。如果信息正确，用户将被引导进入主窗体；否则，系统会弹出提示信息。注册向导窗体模块如图 2.5 所示。

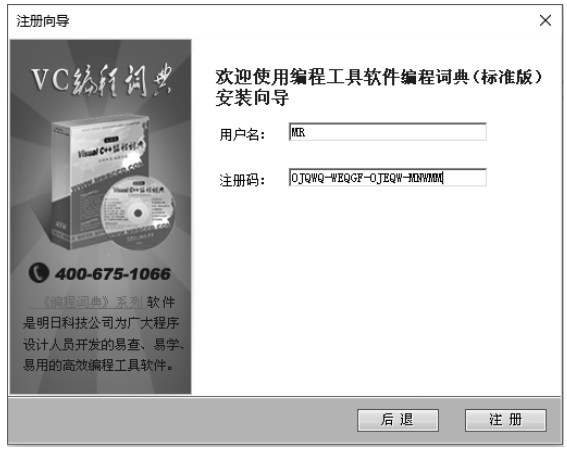


图 2.5 注册向导窗体模块

2.7.2 界面设计

软件注册模块的注册向导窗体界面设计过程如下：

（1）新建一个对话框资源。

（2）向对话框中添加控件，包括两个编辑框控件、两个静态文本控件和两个按钮控件，这些控件的属性设置如表 2.3 所示。

表 2.3 注册向导窗体模块的控件属性设置

控件 ID	控件属性	关联变量
IDC_STATIC	Caption：用户名、Simple	无
IDC_STATIC	Caption：注册码、Simple	无
IDC_EDIT1	无	CString m_Name
IDC_EDIT2	无	CString m_Num1、 CRegEdit m_Edit1
IDC_EDIT3	无	CString m_Num2、 CRegEdit m_Edit2
IDC_EDIT4	无	CString m_Num3、 CRegEdit m_Edit3
IDC_EDIT5	无	CString m_Num4、 CRegEdit m_Edit4
IDC_ADVANCE	Bitmap、Flat	CButton m_Advance
IDC_BACKOFF	Bitmap、Flat	CButton m_Backoff

说明

在为注册码编辑框关联 CString 类型变量时，需要在类向导中将 Maximum Characters 设置为 5，以限制编辑框中输入的字符不能超过 5 个。

2.7.3 设置注册码编辑框

在注册向导窗体模块的 OnInitDialog 方法中，设置注册码编辑框不可用，并注册快捷键。实现代码如下：

```
BOOL CRegisterNumDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    //隐藏注册码编辑框
    GetDlgItem(IDC_EDIT2)->ShowWindow(SW_HIDE);
    //“注册”按钮显示图片
    m_Advance.SetBitmap(LoadBitmap(AfxGetInstanceHandle(),
                                   MAKEINTRESOURCE(IDB_ADVANCE)));
    //“后退”按钮显示图片
    m_Backoff.SetBitmap(LoadBitmap(AfxGetInstanceHandle(),
                                   MAKEINTRESOURCE(IDB_BACKOFF)));
    return TRUE;  //返回 TRUE 以表示成功处理了此消息
}
```

2.7.4 实现“后退”按钮功能

处理“后退”按钮的单击事件，当按钮被按下时，返回注册模块。实现代码如下：

```
void CRegisterNumDlg::OnBackoff()
{
    //调用父类的 OnOk()方法
    CDialog::OnOK();
    //显示模态对话框：选择试用或注册
    CSelectDlg dlg;
    dlg.DoModal();
}
```

2.7.5 实现“注册”按钮功能

处理“注册”按钮的单击事件，当按钮被按下时，首先判断用户输入的用户名和注册码是否正确。如果输入正确，则进行注册；如果输入错误，则提示错误。实现代码如下：

```
void CRegisterNumDlg::OnAdvance()
{
    UpdateData(TRUE);
    //判断用户名和注册码是否已输入
    if(m_Name.IsEmpty() || m_strRegisterCode.IsEmpty()) {
        MessageBox("用户名或注册码错误！");
        return;
    }

    //打开注册表相关键值
    HKEY key;
    char data[32];
    DWORD size = 32;
    DWORD type = REG_SZ;
    CString skey = "Software\\mingrisoft";
    long iret = RegOpenKeyEx(HKEY_CURRENT_USER, skey,
                             REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, &key);
    //打开成功
    if(iret == 0) {
        CString value;
        //查询
        iret = RegQueryValueEx(key, "regnum", 0, &type, (BYTE *)data, &size);
        CString text = data; //AAA-OJQWQWEQGFOJEQWMNWWM
        //查找其中的 '-'
        int index = text.Find("-");
        CString strCode = m_strRegisterCode;
        //删除其中的'-'
        strCode.Replace(_T("-"), _T(""));
        if(iret == 0) {
            //判断是否相等
            if(text.Mid(0, index) == m_Name && text.Mid(1 + index) == strCode) {
                Flag = TRUE;
                RegSetValueEx(key, "isreg", 0, REG_SZ, (BYTE *)"1", 1);
            }
            else {
                MessageBox("用户名或注册码错误！");
                return;
            }
        }
        else {
            RegSetValueEx(key, "regnum", 0, REG_SZ, (BYTE *)"0", 1);
        }
    }
    CDialog::OnOK();
}
```

2.8 项 目 运 行

通过前述步骤，我们设计并完成了“软件注册码生成专家”项目的开发。接下来，我们运行该项目，以检验我们的开发成果。本项目分两部分：一部分是生成注册码，另一部分是软件注册。下面分别介绍在 Visual Studio 2022 中如何运行这两部分。

（1）生成注册码：在项目名称 CreateReg 上右击，选择“设为启动项目”，如图 2.6 所示。然后，在调

试位置选择 Debug、x86，并单击“本地 Windows 调试器”，如图 2.7 所示。

图 2.6　注册生成器项目运行

图 2.7　项目运行

完成这些操作后，即可成功运行注册码功能，并自动打开项目的注册码生成器主窗体，如图 2.8 所示。在“用户名”文本框中输入一个用户名，然后单击“生成注册码”按钮，注册码生成器就会自动生成一串注册码。

（2）注册软件功能：在项目名称 Register 上右击，选择“设为启动项目”，如图 2.9 所示。然后，在调试位置选择 Debug、x86，并单击“本地 Windows 调试器”，如图 2.7 所示。

图 2.8　成功运行项目后进入注册码生成器主窗体

图 2.9　软件注册项目运行

完成这些操作后，即可成功运行软件注册功能，并自动打开项目的软件注册界面，如图 2.10 所示。如果选中“通过注册码注册，使用本软件”单选按钮，系统将自动跳转至输入用户名和注册码界面。如果选中“我想试用此软件”单选按钮，就会自动进入软件注册成功界面。这样，我们就成功地检验了该项目的运行。

图 2.10　软件注册界面

在开发“软件注册码生成专家”项目的过程中，我们重点讲解了两个方面的内容：一是生成注册码，使用系统 API 应用以及硬件信息获取技术实现不同计算机生成唯一注册码；二是如何使用注册表来注册软件功能，包括使用注册码注册软件、免费试用软件（限制 30 次）。通过本章的学习，读者会更深入地掌握系统 API 应用、硬件信息获取以及注册表操作等关键技术，从而在未来开发项目中能够更加得心应手，游刃有余地应对各种挑战。

2.9　源 码 下 载

本章虽然详细地讲解了如何编码实现“软件注册码生成专家”的各个功能，但给出的代码都是代码片段，而非完整的源代码。为了方便读者学习，本书提供了用于下载完整源代码的二维码。

源码下载