

# 实验 1 初步认识 C 语言

## 一、实验目的

- (1) 熟悉最简单的 C 语言程序的基本结构;
- (2) 初步熟悉在 Visual C++ 6.0 环境下编辑、编译和运行 C 语言源程序的过程。

## 二、实验内容及步骤

在启动 Visual C++ 6.0 集成开发环境前,建议在计算机磁盘上(如 D 盘)创建一个新的文件夹,以便存放和管理自己的 C 语言源程序。

### 1、按照下列步骤,完成一个 C 语言源程序的编辑、编译及运行全过程

- (1) 启动 Visual C++ 6.0 集成开发环境

如图 1-1 所示,单击“开始”→“程序”→Microsoft Visual Studio 6.0→Microsoft Visual C++ 6.0,就可以启动 Visual C++ 6.0 集成开发环境。

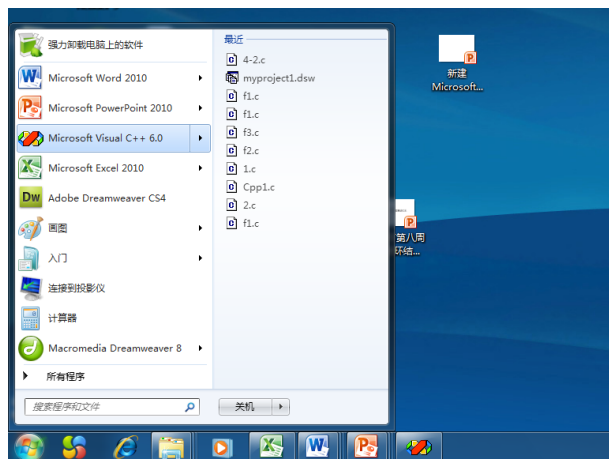


图 1-1 启动 Visual C++ 6.0 的方法

启动后的 Visual C++ 6.0 集成开发环境如图 1-2 所示。

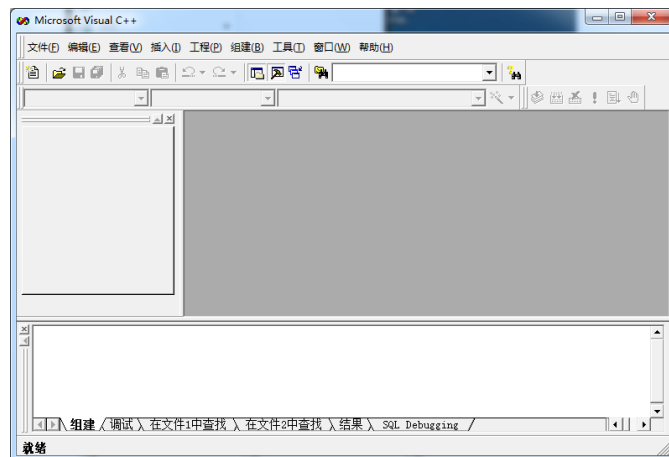


图 1-2 Visual C++ 6.0 中文版集成开发环境

## (2) 开始一个新程序

### ①创建文件

单击主菜单中的“文件”→“新建”菜单命令，弹出“新建”对话框，在“新建”对话框中选择“文件”选项卡。在左边列出的选项中，选择“C++ Source File”；在右边的相应对话框中，输入文件名称“c1-1.c”及保存的位置，如图 1-3 所示。单击“确定”按钮。

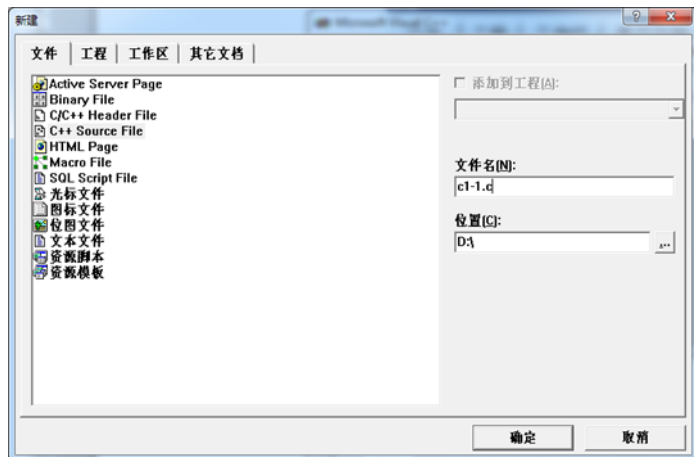


图 1-3 创建新的 C 源文件

进入 Visual C++ 6.0 集成环境的代码编辑窗口，如图 1-4 所示。

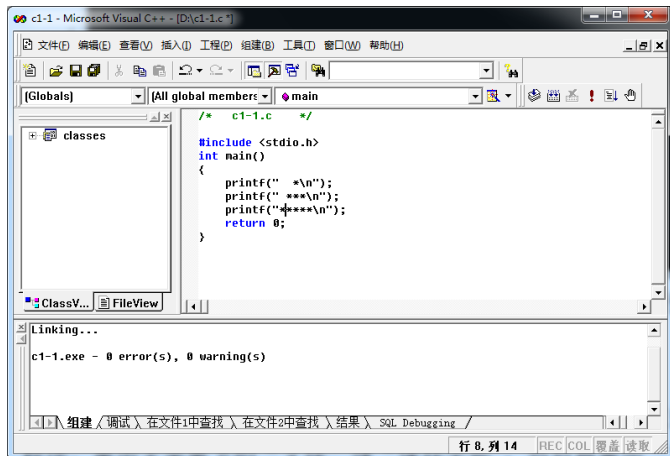


图 1-4 Visual C++ 6.0 集成环境的代码编辑窗口

### ②代码编辑

在 Visual C++ 6.0 代码编辑窗口中，输入如下所示的源代码，完成后如图 1-4 中所示。  
程序代码：

```
/* c1-1.c */
#include <stdio.h>
int main()
{
    printf(" *\\n");
    printf(" ***\\n");
    printf(" *****\\n");
    return 0;
```

}

### ③程序的编译、连接与运行

1) 将 C 语言源代码编译成计算机能执行的目标代码

单击主菜单下的“**组建**”→“**编译 [c1-1.c]**”（或者是工具栏上的按钮或按快捷键 Ctrl+F7），此时将弹出一个对话框，询问是否创建一个项目工作区，选择“是(Y)”。Visual C++ 6.0 集成开发环境会自动在 c1-1.c 文件所在文件夹中建立相应的项目文件。

编译时，在下方的输出框中将显示出相应的编译说明，如图 1-5 所示。

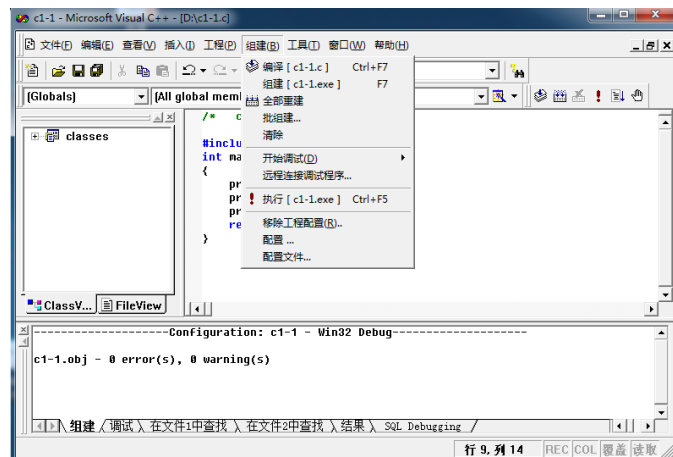


图 1-5 Visual C++ 6.0 集成环境下编译源程序

如果代码编译无误，最后将显示：

c1-1.obj - 0 error(s), 0 warning(s)

这说明编译没有错误（error）和警告（warning），生成目标文件 c1-1.obj，程序编译顺利完成。目标文件（.obj）不能被计算机直接执行，接下来将目标文件（.obj）和相关的库函数或目标程序连接成为可执行程序（.exe）。

单击主菜单下的“**组建**”→“**批组建**”命令，将弹出如图 1-6 所示的对话框。

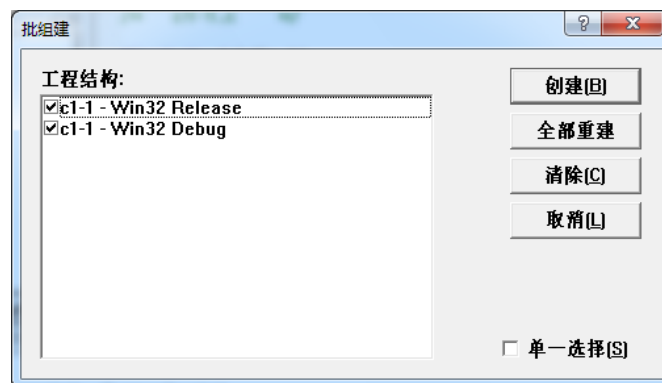


图 1-6 Visual C++ 6.0 集成环境下批组建对话框

确保选中“c1-1-Win32 Release”复选框，这样生成的可执行文件才是发行版的程序，否则生成的是调试（Debug）版的程序。

单击“**创建**”按钮，生成可执行文件 c1-1.exe。如果在“**批组建**”对话框中选中了两个复选框，可以看到程序中生成了两个 c1-1.exe 可执行文件，一个文件为调试版本，存储在与 c1-1.c 同一文件夹下的 Debug 文件夹中；另一个是发行版本，保存在与 c1-1.c 同一文件夹下

这一步只是为了生成发行版的程序文件，只有在程序准备发行时才需要执行这种编译。在通常情况下，可以单击主菜单下的“**组建**”→“**组建 [c1-1.exe]**”（或工具栏按钮或按快捷键 F7），直接生成调试版本程序就可以了。

单击主菜单下的“**组建**”→“**执行[c1-1.exe]**”（或工具栏按钮或按快捷键 Ctrl+F5），此时弹出一个控制台程序窗口，程序正确运行，如图 1-7 所示。按任意键后返回 Visual C++ 6.0 集成开发环境。

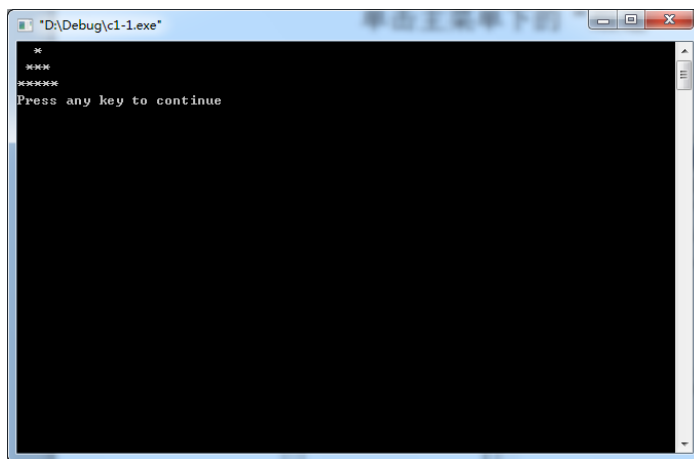


图 1-7 Visual C++ 6.0 集成环境下程序运行结果

当第 1~3 步工作完成后,应该将工作保存下来,并关闭工作空间,以便做下一个新的程序。单击主菜单下的“文件”→“保存全部”,然后再单击“文件”→“关闭工作空间”菜单命令,询问确认要关闭所有文档窗口,选择“是(Y)”。

在打开已有的文件时一定要先确认关闭了工作空间，然后单击主菜单下的“文件”→“打开”命令，选择创建的 **c1-1.c** 文件，然后打开。

2、根据以上介绍的利用 VC++6.0 集成环境建立编译及运行程序的方法,练习建立、编译及运行以下程序 **c1-2.c**。该程序运行后将输出一个同学的自我介绍。

```

/*  c1-2.c  */
#include<stdio.h>

int main()
{
    printf("*****\n");
    printf("        ^_^ Hello,大家好! ^_^        \n");
    printf("我是计算机科学与技术专业 2021051 班的学生\n");
    printf("        我叫章小龙        \n");
    printf("        很高兴认识大家!        \n");
    printf("*****\n");
    return 0;
}

```

① 在集成环境下对程序进行编辑、调试、运行。这个程序的运行结果如图 1-8 所示。

```
*****
^_^ Hello, 大家好! ^_^
我是计算机科学与技术专业2021051班的学生
我叫章小龙
很高兴认识大家!
*****
```

图 1-8 程序 c1-2.c 的运行结果

② 修改程序 c1-2.c 的输出格式，使输出结果如图 1-9，程序另存为 c1-3.c。

```
*****
*      ^_^ Hello, 大家好! ^_^      *
* 我是计算机科学与技术专业2021051班的学生 *
*      我叫章小龙      *
*      很高兴认识大家!      *
*****
```

图 1-9 程序 c1-3.c 的输出结果

③ 将程序 c1-2.c 中的相关专业、班级、姓名内容改成同学本人的相关信息，程序另存为 c1-4.c。

## 实验 2 基本输入输出

### 一、实验目的

- (1) 熟练掌握在 Visual C++ 6.0 环境下编辑、编译和运行 C 源程序的步骤和方法；
- (2) 掌握基本数据类型的概念，各种类型数据变量的定义和赋值方法；
- (3) 掌握各种类型数据的输入输出的方法，并能正确使用各种数据类型的输出输入格式控制符。

### 二、实验内容及步骤

#### I. 基础部分：理解基本输入输出语句的使用

关闭所有工作空间，点击 Visual C++ 6.0 窗口的关闭按钮即可退出 Visual C++ 6.0 集成环境，按照以上步骤再次进入 Visual C++ 6.0 集成环境，输入以下程序，运行并查看输出结果。

(1) 字符串的输出及换行符的使用

① 编辑、调试并运行程序 c2-1-1.c。

```
/* c2-1-1.c */
#include <stdio.h>
```

```
int main()
{
    printf("春眠不觉晓");
    printf("处处闻啼鸟");
    printf("夜来风雨声");
    printf("花落知多少");
    return 0;
}
```

调试运行程序，该程序的运行结果如图 2-1:

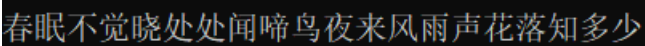


图 2-1 程序 c2-1-1.c 的运行结果

从运行结果可以看出，程序的输出结果不尽如人意。

- ② 将源程序 c2-1-1.c 稍作修改，每个输出语句的最后增加一个换行，程序另存为 c2-1-2.c。

```
/* c2-1-2.c */
#include <stdio.h>
int main()
{
    printf("春眠不觉晓\n");
    printf("处处闻啼鸟\n");
    printf("夜来风雨声\n");
    printf("花落知多少\n");
    return 0;
}
```

这个程序的运行结果如图 2-2 所示。

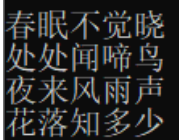


图 2-2 程序 c2-1-2.c 的运行结果

- ③ 进一步修改程序 c2-1-2.c，另存为 c2-1-3.c，得到如图 2-3 所示运行结果。

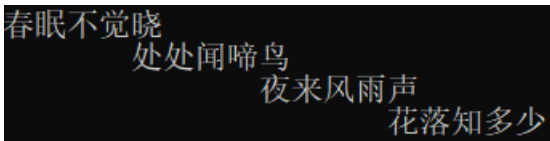


图 2-3 程序 c2-1-3.c 的运行结果

## (2) 整型数据变量的定义、赋值和输入输出

- ① 编辑、调试并运行程序 c2-1-4.c。

```
/* c2-1-4.c */
#include <stdio.h>
int main()
{
    int x,y,sum;                //定义整型变量
```

```

x=5;                //为整型变量 x 赋值
y=6;                //为整型变量 y 赋值
sum=x+y;
printf("x+y=%d",sum); //输出整型变量 sum 的值
return 0;
}

```

②修改程序，为整型变量 x、y 赋值方式改成从键盘输入，即通过 scanf() 函数输入，重新调试该程序，源程序命名为 c2-1-5.c。

```

/* c2-1-5.c */
#include <stdio.h>
int main()
{
    int x,y,sum;        //定义整型变量
    printf("Input x and y:\n");
    scanf("%d",&x);      //从键盘输入为整型变量 x 的值
    scanf("%d",&y);      //从键盘输入整型变量 y 的值
    sum=x+y;
    printf("x+y=%d",sum); //输出整型变量 sum 的值
    return 0;
}

```

③修改程序 c2-1-5.c，通过一个 scanf() 函数实现键盘输入变量 x、y 的值，源程序命名为 c2-1-6.c。分析在输入格式上和 c2-1-5.c 有没有区别，为什么？

```

/* c2-1-6.c */
#include <stdio.h>
int main()
{
    int x,y,sum;        //定义整型变量
    printf("Input x and y:\n");
    scanf("%d%d",&x,&y); //从键盘输入为整型变量 x、y 的值
    sum=x+y;
    printf("x+y=%d",sum); //输出整型变量 sum 的值
    return 0;
}

```

④修改程序 c2-1-6.c，在 scanf() 函数中格式符的 %d%d 中间加“，”，源程序命名为 c2-1-7.c。分析在输入格式上和 c2-1-6.c 有没有区别，为什么？

```

/* c2-1-7.c */
#include <stdio.h>
int main()
{
    int x,y,sum;
    printf("Input x and y:\n");
    scanf("%d,%d",&x,&y); //输入格式有变
    sum=x+y;
}

```

```

        printf("x+y=%d",sum);
        return 0;
}

```

- ⑤ 有同学喜欢把 scanf()函数写成如下格式:

```
scanf("x=%d,y=%d",&x,&y);
```

这样的写法在语法上是认可的, 在输入数据时应该如何输入?

```

/*    c2-1-8.c    */
#include <stdio.h>
int  main()
{
    int  x,y,sum;
    printf("Input x and y:\n");
    scanf("x=%d,y=%d",&x,&y);    //输入格式有变
    sum=x+y;
    printf("x+y=%d",sum);
    return 0;
}

```

### (3) 单精度浮点数数据变量的定义、赋值和输入输出

- ① 编辑、调试并运行程序 c2-1-9.c。

```

/*    c2-1-9.c    */
#include <stdio.h>
int  main()
{
    float  fa,fb;                //定义单精度型变量
    fa=3141.592678;              //为单精度型变量 fa 赋值
    fb=123.4;                    //为单精度型变量 fb 赋值
    printf("float 类型数据的打印结果: \n");
    printf("fa=%f\t fb=%f\n",fa,fb);    //输出单精度型变量 fa 和 fb 的值
    return  0;
}

```

- ② 将该程序的变量 fa, fb 的赋值方式改成从键盘输入, 即通过 scanf () 函数输入, 重新调试该程序, 源程序命名为 c2-1-10.c。

```

/*    c2-1-10.c    */
#include <stdio.h>
int  main()
{
    float  fa,fb;
    printf("Input x and y:\n");
    scanf("%f%f",&fa,&fb);    //从键盘输入 fa、fb 的值
    printf("float 类型数据的打印结果: \n");
    printf("fa=%f\t fb=%f\n",fa,fb);
    return  0;
}

```



```
}
```

③ 改变 printf() 中的输出格式，程序另存为 c2-1-11.c，观察在输出格式上：a) 和 c2-1-10.c 的区别，b) 变量 fa 和 fb 格式上的区别。

```
/* c2-1-11.c */
#include <stdio.h>
int main()
{
    float fa,fb;
    printf("Input x and y:\n");
    scanf("%f%f",&fa,&fb); //从键盘输入 fa、fb 的值
    printf("float 类型数据的打印结果: \n");
    printf("fa=%10.3f\t fb=%10.3f\n",fa,fb);
    return 0;
}
```

④ 浮点型数据对应的输入输出格式符是 %f，如果在 scanf() 函数中，将格式符误用为 %d，如程序 c2-1-12.c，结果会如何？

```
/* c2-1-12.c */
#include <stdio.h>
int main()
{
    float fa,fb;
    printf("Input x and y:\n");
    scanf("%d%d",&fa,&fb); //变量 fa、fb 对应的格式符错误
    printf("float 类型数据的打印结果: \n");
    printf("fa=%10.3f\t fb=%10.3f\n",fa,fb);
    return 0;
}
```

⑤ 如果在 printf() 函数中，将浮点型数据的格式符误用为 %d，如程序 c2-1-13.c，结果又会如何？

```
/* c2-1-13.c */
#include <stdio.h>
int main()
{
    float fa,fb;
    printf("Input x and y:\n");
    scanf("%f%f",&fa,&fb);
    printf("float 类型数据的打印结果: \n");
    printf("fa=%d\t fb=%d\n",fa,fb); //变量 fa、fb 对应的格式符错误
    return 0;
}
```

(4) 字符数据类型变量的定义、赋值和输入输出

① 编辑、调试并运行程序 c2-1-14.c，分析为什么 c1 和 c2 输出的结果形式不同。

```
/* c2-1-14.c */
#include <stdio.h>
```

```

int main()
{
    char c1,c2;           //定义字符型变量
    c1='A';               //为字符型变量 c1 赋值
    c2='B';               //为字符型变量 c2 赋值
    printf("c1=%c;c2=%d",c1,c2); //输出字符型变量 c1 和 c2 的值
    return 0;
}

```

- ② 将程序中变量 c1、c2 改成从键盘通过 scanf() 函数输入，重新调试该程序，源程序命名为 c2-1-15.c。输入数据时，a) 输入的 2 个字符之间没有空格，b) 输入的字符之间有空格，分别分析这两种输入得到的不同的输出结果。

```

/* c2-1-15.c */
#include <stdio.h>
int main()
{
    char c1,c2;
    printf("Input c1 and c2:\n");
    scanf("%c%c",&c1,&c2); //从键盘输入变量 c1、c2 的值
    printf("c1=%c;c2=%c",c1,c2); //输出字符型变量 c1 和 c2 的值
    return 0;
}

```

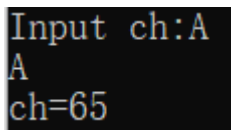
## II 提高部分：格式化输入、输出语句的使用

### 1、程序填空

说明：程序有多个空 (1)、(2) ……需要补充完整。请将程序中的\_\_ (1) \_\_、\_\_ (2) \_\_……删除后，在相应的位置填入正确答案并调试直到得到正确结果为止。注意：不要随意改动程序，不得增行或删行，也不得更改程序的结构！

(1) 程序 c2-2-1.c 所实现的功能是：输入一个大写字母（例如字符 'A'），将它转换为对应的小写字母，输出小写字母及对应的 ASCII 码值。请完善程序。

程序运行结果如下：



```

Input ch:A
A
ch=65

```

图 2-4 程序 c2-2-1.c 的运行结果

```

/* c2-2-1.c */
#include <stdio.h>
int main()
{
    int j;
    char ch;
    printf("Input ch:");
    /**found***/
}

```

```

    ____ (1) ____;      //用 getchar()函数从键盘输入变量 ch 的值
    j=ch+32;           //大小写 ASCII 码值相差 32
    /*****found*****/

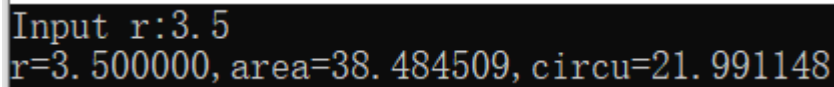
    ____ (2) ____;      //用 putchar()函数输出变量 ch 的值
    /*****found*****/

    ____ (3) ____;      //用 printf()函数输出变量 ch 的 ASCII 值
    return 0;
}

```

(2) 程序 c2-2-2.c 所实现的功能是：从键盘输入圆半径，计算圆面积和圆周长，并输出。请完善程序。

程序运行结果示例：



```

Input r:3.5
r=3.500000, area=38.484509, circu=21.991148

```

图 2-5 程序 c2-2-2.c 的运行结果

```

/* c2-2-2.c */
#include <stdio.h>
int main( )
{
    double r, area, circu; //圆的半径、面积、周长
    printf("Input r:");
    /*****found*****/

    ____ (1) ____;      //键盘输入变量 r 的值，注意 r 的数据类型是 double
    area=3.1415926*r*r;
    circu=2*3.1415926*r;
    /*****found*****/

    ____ (2) ____;      //输出 r、area、circu 的值
    return 0;
}

```

### III.拓展部分：文本文件的使用

(1) 输入并调试下列程序，程序保存为 c2-3-1.c。

```

/* c2-3-1.c */
#include <stdio.h>
int main()
{
    FILE *fp;
    //以写的方式打开文件 test.txt，默认文件路径和源程序文件 c2-11.c 相同
    fp=fopen("test.txt", "w");
    printf("我爱你，中国！"); //输出到屏幕
    fprintf(fp, "我爱你，中国！"); //输出到文件
    fclose(fp);
}

```

```
return 0;  
}
```

调试并运行程序，屏幕输出结果如下：

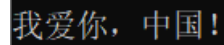


图 2-6 程序 c2-3-1.c 的运行结果

在和 c2-3-1.c 相同的文件夹中找到文件 test.txt，打开文件，文件内容如下：

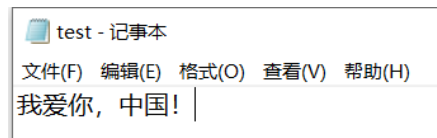


图 2-5 程序运行后文本文件“test.txt”的内容

(2) 编写程序 c2-3-2.c，将古诗“春晓”在屏幕上输出的同时，输出到文件“春晓.txt”中。

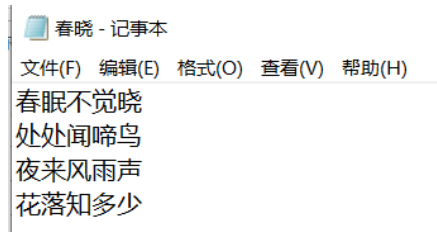


图 2-6 程序运行后文本文件“春晓.txt”的内容