计算机程序设计

第 1 章 C 语言概述

Jinlong Li[1]

jlli@ustc.edu.cn

[1]School of Computer Science

University of Science and Technology of China

July 22, 2011

Outline of Topics

C 语言的简单历史回顾

- created by Dennis Ritchie at the Bell Telephone Laboratories in 1972;
- purpose: to design the UNIX operating system;
- American National Standards Institute (ANSI) formed a committee in 1983 to establish a standard definition of C;
- ANSI-89/90.

C 语言的简单历史回顾

- created by Dennis Ritchie at the Bell Telephone Laboratories in 1972;
- purpose: to design the UNIX operating system;
- American National Standards Institute (ANSI) formed a committee in 1983 to establish a standard definition of C;
- ANSI-89/90.

C 语言的简单历史回顾

- created by Dennis Ritchie at the Bell Telephone Laboratories in 1972;
- purpose: to design the UNIX operating system;
- American National Standards Institute (ANSI) formed a committee in 1983 to establish a standard definition of C;
- ANSI-89/90.

C 语言的简单历史回顾

- created by Dennis Ritchie at the Bell Telephone Laboratories in 1972;
- purpose: to design the UNIX operating system;
- American National Standards Institute (ANSI) formed a committee in 1983 to establish a standard definition of C;
- ANSI-89/90.

为什么用 C 语言

C 语言的优点和缺点

- C 语言是一种功能强大，灵活方便的程序设计语言；
- C 语言是最流行的专业程序设计语言；
- C 语言可移植性好，能方便地在不同硬件、软件系统中编译和运行；
- 关键词较少；
- 面向过程的模块化程序设计语言。

C++ 是面向对象的程序设计语言，是 C 语言的超集；学会了 C 语言，再学习其他语言如：C++，Java，Python，php，Perl，Javascript，C# 等会非常容易。

为什么用 C 语言

C 语言的优点和缺点

- C 语言是一种功能强大，灵活方便的程序设计语言；
- C 语言是最流行的专业程序设计语言；
- C 语言可移植性好，能方便地在不同硬件、软件系统中编译和运行；
- 关键词较少；
- 面向过程的模块化程序设计语言。

C++ 是面向对象的程序设计语言，是 C 语言的超集；学会了 C 语言，再学习其他语言如：C++，Java，Python，php，Perl，Javascript，C# 等会非常容易。

为什么用 C 语言

C 语言的优点和缺点

- C 语言是一种功能强大，灵活方便的程序设计语言；
- C 语言是最流行的专业程序设计语言；
- C 语言可移植性好，能方便地在不同硬件、软件系统中编译和运行；
- 关键词较少；
- 面向过程的模块化程序设计语言。

C++ 是面向对象的程序设计语言，是 C 语言的超集；学会了 C 语言，再学习其他语言如：C++，Java，Python，php，Perl，Javascript，C# 等会非常容易。

为什么用 C 语言

C 语言的优点和缺点

- C 语言是一种功能强大，灵活方便的程序设计语言；
- C 语言是最流行的专业程序设计语言；
- C 语言可移植性好，能方便地在不同硬件、软件系统中编译和运行；
- 关键词较少；
- 面向过程的模块化程序设计语言。

C++ 是面向对象的程序设计语言，是 C 语言的超集；学会了 C 语言，再学习其他语言如：C++, Java，Python，php, Perl, Javascript, C# 等会非常容易。

为什么用 C 语言

C 语言的优点和缺点

- C 语言是一种功能强大，灵活方便的程序设计语言；
- C 语言是最流行的专业程序设计语言；
- C 语言可移植性好，能方便地在不同硬件、软件系统中编译和运行；
- 关键词较少；
- 面向过程的模块化程序设计语言。

C++ 是面向对象的程序设计语言，是 C 语言的超集；学会了 C 语言，再学习其他语言如：C++，Java，Python，php，Perl，Javascript，C# 等会非常容易。

Outline

基本步骤

采用不同程序设计语言编程都有类似的步骤

- 明确程序要完成的任务或目标；
- 确定为完成任务而采用的解决办法；
- 编写程序来解决问题；
- 运行程序，检查结果。

基本步骤

采用不同程序设计语言编程都有类似的步骤

- 明确程序要完成的任务或目标；
- 确定为完成任务而采用的解决办法；
- 编写程序来解决问题；
- 运行程序，检查结果。

基本步骤

采用不同程序设计语言编程都有类似的步骤

- 明确程序要完成的任务或目标；
- 确定为完成任务而采用的解决办法；
- 编写程序来解决问题；
- 运行程序，检查结果。

基本步骤

采用不同程序设计语言编程都有类似的步骤

- 明确程序要完成的任务或目标；
- 确定为完成任务而采用的解决办法；
- 编写程序来解决问题；
- 运行程序，检查结果。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法后，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本，notepad；
  - MS Word；
  - Vi, Emacs；
  - 集成开发环境(IDE)，如 Visual Studio, Borland C++ 等；
  - 其他任何文本编辑器都可以。
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法后，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本，notepad；
  - MS Word；
  - Vi, Emacs；
  - 集成开发环境(IDE)，如 Visual Studio, Borland C++ 等；
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE)，如 Visual Studio, Borland C++ 等；
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

编写源程序

源代码和编辑器

- 有了明确的目标和解决的办法，
- 需要用一种语言表述解决办法： C 语言；
- 需要用一种工具记录这种表述：纸笔--->编辑器；
  - 记事本,notepad;
  - MS Word;
  - Vi, Emacs;
  - 集成开发环境(IDE), 如 Visual Studio, Borland C++ 等;
  - 其他任何文本编辑器都可以.
- 用 C 语言表述了解决办法后，用编辑器记录下来，保存成文件，这个文件称为“源代码”文件。我们的课程将会教授大家：
  - 如何创建 C 语言源文件：语言基本组成、语法，结构等等；
  - 如何验证其正确性：编译进行语法检查，调试进行逻辑检查；
  - 如何组织源文件：大型项目包括很多的源文件，如何使得源代码有条理，便于查看。

第一个程序的源代码

Hello.c

```
1 #include <stdio.h>
2 int main(){
3     printf(``Hello!'');
4 }
```

- 任务目标： 显示欢迎信息“hello!”；
- 解决方法：利用已有的库函数 printf() 直接输出欢迎信息；
- 源代码文件是解决方案的描述，并不是可执行的程序，我们需要一个转化操作，把这个源代码文件转换为可执行的程序；
- 这个转换称为：“编译”，注意和另两个词的区别：编辑，翻译。

第一个程序的验证

编译：检查语法，生成可执行代码/可执行文件

- 集成开发环境（IDE）：从菜单选择“编译”命令或使用快捷键；
- 利用 Makefile，用 make 在命令行进行编译；
- 不利用任何工具，直接在命令行运行编译命令：
  - Linux/Unix: g++,gcc,cc 等
  - Windows 下微软 c: cl.exe
  - Turbo C: tcc.exe
  - Borland C: bcc.exe

```
1  gcc hello.c   //将会产生一个 a.out 文件
2             //这就是转换后得到的可执行程序
```

Gcc 编译过程

包括四步，为了得到可执行的程序，这四步都会进行

1. 预编译：-E 不编译、不汇编和不链接，
2. 编译（compile）：-S 预编译 + 编译，但不汇编、不链接；
3. 汇编（assemble）：-c 预编译 + 编译 + 汇编，但是不链接；
4. 链接（link）：-o <file> 预编译 + 编译 + 汇编 + 链接，并将可执行程序的名字命名为<file>

```
1  gcc -E hello.c  //仅进行预编译
2   //处理 #include <stdio.h>语句
3  gcc -S hello.c  //进行编译，得到 hello.s 文件
4                  //汇编代码，一种低级编程语言
5  gcc -c hello.c  //进行编译、汇编，得到
6             //hello.o 文件，二进制代码，机器语言
7  gcc -o hello hello.c  //进行编译、汇编和链
8       //接得到 hello 文件，二进制代码，可执行程序
```

Outline

编译错误

语法错误信息格式

源代码程序文件 b.cpp 如下：

```
1  int main(){
2      printf( “hello! ” ) }
```

运行命令编译上面源代码：gcc b.cpp ，显示编译错误信息如下：

```
1  b.cpp: In function ‘int main()’ :
2  b.cpp:5:17: error: 'printf' was not
       declared in this scope
3  b.cpp:6:1: error: expected ';' before '}'
       token
```

1. 第 1 行指出 b.cpp 文件中 main()函数有语法错，错误如下：
2. 第 2 行指出 b.cpp 文件的第 5 行 17 列出错，printf 没有定义；
3. 第 3 行指出 b.cpp 文件的第 6 行第 1 列出错，在}的前面应该有个分号

编译错误

警告信息格式

```
1 b.cpp: In function ‘int main()’ :
2 b.cpp:5:28: warning: format ‘%’ d
    expects type ‘’ int, but argument
    2 has type ‘’ double
```

1. 第 1 行指出出问题的地方在 b.cpp 文件的 main()函数内；
2. 第 2 行指出这是一个警告信息(warning)，警告不同与错误（error），警告一般不会停止编译的后续步骤，如果仅有警告信息，编译会完成指定的所有编译步骤；而编译中遇到错误时，会终止编译过程；
3. 注意，在编译警告信息显示时，还会生成可执行程序，这表示程序存在可能存在一些非致命错误（fatal error），这种警告意味着编译器认为错误不是很严重，不会影响到生成可执行程序；但是这只是编译器的看法，实际上有可能这些警告会暗示有逻辑错误在里面，会造成程序执行结果的完全错误。
4. 例子中的警告信息指出，参数 2 的类型是 double，但是输出的格式是 %d，即整数，也就是这个函数要把一个 double 类型的实数当成整数打印出来，打印的结果会出错。

总结和问题

- 编程的基本概念和术语：源代码、可执行程序、编译、编辑等等；
- C 程序源代码的后缀名是什么？
- gcc 编译器将编译分为哪几个部分？
- 编译错误信息的阅读和理解！！！！

总结和问题

- 编程的基本概念和术语：源代码、可执行程序、编译、编辑等等；
- C 程序源代码的后缀名是什么？
- gcc 编译器将编译分为哪几个部分？
- 编译错误信息的阅读和理解！！！！

总结和问题

- 编程的基本概念和术语：源代码、可执行程序、编译、编辑等等；
- C 程序源代码的后缀名是什么？
- gcc 编译器将编译分为哪几个部分？
- 编译错误信息的阅读和理解！！！！

总结和问题

- 编程的基本概念和术语：源代码、可执行程序、编译、编辑等等；
- C 程序源代码的后缀名是什么？
- gcc 编译器将编译分为哪几个部分？
- 编译错误信息的阅读和理解！！！！

练习 1

阅读程序:这个程序有什么用？

```
1  #include <stdio.h>
2
3  int radius, area;
4
5  main(){
6      printf( "Enter␣radius␣(i.e.␣10):␣
           " );
7      scanf( "%d", &radius );
8      area = (int) (3.14159 * radius *
           radius);
9      printf( "\n\nArea␣=␣%d\n", area )
           ;
10     return 0;
11 }
```

练习 2

阅读程序： 程序做什么事情？

```
1  #include <stdio.h>
2
3  int x,y;
4
5  main()
6  {
7      for ( x = 0; x < 10; x++, printf(
           "\n" ))
8          for ( y = 0; y < 10; y++ )
9              printf( "X" );
10
11     return 0;
12 }
```

练习 2

自己编译，理解出错信息，设法改正

```c
#include <stdio.h>
 main(); {
      printf( "Keep looking!" );
      printf( "You\'ll find it!\n" );
      return 0;
 }
```

```c
#include <stdio.h>
 main() {
      printf( "This is a program with
          a " );
      do_it( "problem!");
      return 0;
 }
```

Outline

典型的例子

五脏俱全的 C 语言程序

```
1  /* to calculate the product of two
       numbers*/
2  #include <stdio.h>
3  int a,b,c;
4  int product(int x, int y);
5  main()
6  {
7      /* Input the first number */
8      printf("Enter␣a␣number␣between␣1
           ␣and␣100:␣");
9      scanf("%d", &a);
10     /* Input the second number */
11     printf("Enter␣another␣number␣
           between␣1␣and␣100:␣");
```

典型的例子

五脏俱全的 C 语言程序：续

```
12      scanf("%d", &b);
13
14      /* Calculate and display the
           product */
15      c = product(a, b);
16      printf ("%d␣times␣%d␣=␣%d\n", a,
            b, c);
17      return 0;
18  }
19
20  /* Function returns the product of
       its two arguments */
21  int product(int x, int y) {
22      return (x * y);
23  }
```

Outline

第一个组成部分

main()函数：程序第 5 行到第 18 行

- 一个 C 程序中必须要包含的内容：main()函数；
- 唯一一个必须要包含的函数；
- 最简单的 main()函数：main(){}
  这就是最简单的程序
  花括号内放置程序主体，要完成的工作都放在花括号内；
- 一般情况下：程序执行时，执行 main()函数的第一条语句；程序结束时执行 main()函数的最后一条语句；
- 特殊情形时，程序可能不是从 main()函数的第一句开始执行，或者不是从最后一条语句退出。

第一个组成部分

main()函数：程序第 5 行到第 18 行

- 一个 C 程序中必须要包含的内容：main()函数；
- 唯一一个必须要包含的函数；
- 最简单的 main()函数：main(){}
  这就是最简单的程序
  花括号内放置程序主体，要完成的工作都放在花括号内；
- 一般情况下：程序执行时，执行 main()函数的第一条语句；程序结束时执行 main()函数的最后一条语句；
- 特殊情形时，程序可能不是从 main()函数的第一句开始执行，或者不是从最后一条语句退出。

第一个组成部分

main()函数：程序第 5 行到第 18 行

- 一个 C 程序中必须要包含的内容：main()函数；
- 唯一一个必须要包含的函数；
- 最简单的 main()函数：main(){}
  这就是最简单的程序
  花括号内放置程序主体，要完成的工作都放在花括号内；
- 一般情况下：程序执行时，执行 main()函数的第一条语句；程序结束时执行 main()函数的最后一条语句；
- 特殊情形时，程序可能不是从 main()函数的第一句开始执行，或者不是从最后一条语句退出。

第一个组成部分

main()函数：程序第 5 行到第 18 行

- 一个 C 程序中必须要包含的内容：main()函数；
- 唯一一个必须要包含的函数；
- 最简单的 main()函数：main(){}
  这就是最简单的程序
  花括号内放置程序主体，要完成的工作都放在花括号内；
- 一般情况下：程序执行时，执行 main()函数的第一条语句；程序结束时执行 main()函数的最后一条语句；
- 特殊情形时，程序可能不是从 main()函数的第一句开始执行，或者不是从最后一条语句退出。

第一个组成部分

main()函数：程序第 5 行到第 18 行

- 一个 C 程序中必须要包含的内容：main()函数；
- 唯一一个必须要包含的函数；
- 最简单的 main()函数：main(){}
  这就是最简单的程序
  花括号内放置程序主体，要完成的工作都放在花括号内；
- 一般情况下：程序执行时，执行 main()函数的第一条语句；程序结束时执行 main()函数的最后一条语句；
- 特殊情形时，程序可能不是从 main()函数的第一句开始执行，或者不是从最后一条语句退出。

第二个组成部分

#include 命令

- #include 命令告诉编译器，将该命令指定的文件内容添加到本源代码文件内（在编译之前添加，用于编译）；
- 这个指定文件我们称为“include 文件”，包含了程序或编译器需要的信息或代码；
- 编译器需要的这些文件，有时我们称为“头文件/header files”，这些文件不需要作出任何修改，一般具有后缀名.h;
- 例子：下面两个 include 命令告诉编译器在编译的时候，把 stdio.h 和 hello.h 两个文件的内容包含到源代码文件中去.

```
1 #include <stdio.h>
2 #include "hello.h"
```

第三个组成部分

变量定义：程序第 3 行

```
1  int a,b,c;
```

- 定义了三个全局变量 a,b,c，这三个全局变量能存放整数数据；
- C 语言中任何变量在使用之前一定都要先定义，所谓定义就是指出变量的类型；
- 而变量定义所在的位置，决定了变量是全局的，还是局部的，即：作用范围，变量有效范围；
- 所谓变量就是计算机内存的某些字节的“名称”。

第四个组成部分

函数原型说明：程序第 4 行

```
1    int product(int x, int y);
```

- 在任何一个函数被使用之前，编译器需要知道这个函数是否已经被定义了；
- 函数原型说明告诉编译器：“已经在其他地方定义了函数 product()，这个函数有两个整数类型的输入变量，返回值是一个整数”．

第五个组成部分

程序执行语句：第 8,9,10,12,15,16,17 行

- 计算机程序的实际工作由程序执行语句完成；
- 程序执行语句包括：从键盘或鼠标读数据，向文件或显示器写数据，调用函数，进行算术运算，读文件等等；
- C 语言的语句标志：分号“;”表示一个语句的结束，同时也可能是下一条语句的开始；
- C 语言大小写敏感：即变量、函数或语句中大小写不一样所代表的含义不一样.

第五个组成部分

程序执行语句：第 8,9,10,12,15,16,17 行

- 计算机程序的实际工作由程序执行语句完成；
- 程序执行语句包括：从键盘或鼠标读数据，向文件或显示器写数据，调用函数，进行算术运算，读文件等等；
- C 语言的语句标志：分号“;”表示一个语句的结束，同时也可能是下一条语句的开始；
- C 语言大小写敏感：即变量、函数或语句中大小写不一样所代表的含义不一样.

第五个组成部分

程序执行语句：第 8,9,10,12,15,16,17 行

- 计算机程序的实际工作由程序执行语句完成；
- 程序执行语句包括：从键盘或鼠标读数据，向文件或显示器写数据，调用函数，进行算术运算，读文件等等；
- C 语言的语句标志：分号“;”表示一个语句的结束，同时也可能是下一条语句的开始；
- C 语言大小写敏感：即变量、函数或语句中大小写不一样所代表的含义不一样.

第五个组成部分

程序执行语句：第 8,9,10,12,15,16,17 行

- 计算机程序的实际工作由程序执行语句完成；
- 程序执行语句包括：从键盘或鼠标读数据，向文件或显示器写数据，调用函数，进行算术运算，读文件等等；
- C 语言的语句标志：分号“;”表示一个语句的结束，同时也可能是下一条语句的开始；
- C 语言大小写敏感：即变量、函数或语句中大小写不一样所代表的含义不一样.

第六个组成部分

函数定义：第 21 ~ 23 行

- 编程至少需要定义一个函数：main( ) 函数；
- 上述例子定义了一个非 main()函数，其定义方法基本类同于 main()函数的定义；
- 所谓函数就是能独立完成一定工作的语句集合；每个函数有个名字（函数名），我们可以通过函数名来调用（访问）这个函数；
- 所谓函数调用就是在一个函数中有一个语句，这个语句告知要执行另一个函数（所包含的语句集合）；
- C 语言的程序最终会被组织称为一个个具有独立、简单功能的函数，这些函数之间用某种规律相互调用，组合成复杂的功能，进而完成指定的任务。

第六个组成部分

函数定义：第 21 ~ 23 行

- 编程至少需要定义一个函数：main( ) 函数；
- 上述例子定义了一个非 main()函数，其定义方法基本类同于 main()函数的定义；
- 所谓函数就是能独立完成一定工作的语句集合；每个函数有个名字（函数名），我们可以通过函数名来调用（访问）这个函数；
- 所谓函数调用就是在一个函数中有一个语句，这个语句告知要执行另一个函数（所包含的语句集合）；
- C 语言的程序最终会被组织称为一个个具有独立、简单功能的函数，这些函数之间用某种规律相互调用，组合成复杂的功能，进而完成指定的任务。

第六个组成部分

函数定义：第 21 ~ 23 行

- 编程至少需要定义一个函数：main( ) 函数；
- 上述例子定义了一个非 main()函数，其定义方法基本类同于 main()函数的定义；
- 所谓函数就是能独立完成一定工作的语句集合；每个函数有个名字（函数名），我们可以通过函数名来调用（访问）这个函数；
- 所谓函数调用就是在一个函数中有一个语句，这个语句告知要执行另一个函数（所包含的语句集合）；
- C 语言的程序最终会被组织称为一个个具有独立、简单功能的函数，这些函数之间用某种规律相互调用，组合成复杂的功能，进而完成指定的任务。

第六个组成部分

函数定义：第 21 ~ 23 行

- 编程至少需要定义一个函数：main( ) 函数；
- 上述例子定义了一个非 main()函数，其定义方法基本类同于 main()函数的定义；
- 所谓函数就是能独立完成一定工作的语句集合；每个函数有个名字（函数名），我们可以通过函数名来调用（访问）这个函数；
- 所谓函数调用就是在一个函数中有一个语句，这个语句告知要执行另一个函数（所包含的语句集合）；
- C 语言的程序最终会被组织称为一个个具有独立、简单功能的函数，这些函数之间用某种规律相互调用，组合成复杂的功能，进而完成指定的任务。

第六个组成部分

函数定义：第 21 ~ 23 行

- 编程至少需要定义一个函数：main( ) 函数；
- 上述例子定义了一个非 main()函数，其定义方法基本类同于 main()函数的定义；
- 所谓函数就是能独立完成一定工作的语句集合；每个函数有个名字（函数名），我们可以通过函数名来调用（访问）这个函数；
- 所谓函数调用就是在一个函数中有一个语句，这个语句告知要执行另一个函数（所包含的语句集合）；
- C 语言的程序最终会被组织称为一个个具有独立、简单功能的函数，这些函数之间用某种规律相互调用，组合成复杂的功能，进而完成指定的任务。

第七个组成部分

注释：/*.....*/和//

- 注释就是在程序源代码文件中添加说明性文字，方便记忆、阅读；
- 程序代码写了后，时间长了，就会忘记其含义，当再次阅读或别人阅读，要进行修改时，注释会极大地提高效率；
- C 语言的注释有两种/* 和/* 括起来的文字，可以跨行，都被视为注释内容；
- 也可以用简单方便的//来注释当前行从//开始的文字；
- 注释后的文字会被编译器忽略掉，意味着注释的文字不会对程序的功能或执行产生任何影响。

第七个组成部分

注释：/*.....*/和//

- 注释就是在程序源代码文件中添加说明性文字，方便记忆、阅读；
- 程序代码写了后，时间长了，就会忘记其含义，当再次阅读或别人阅读，要进行修改时，注释会极大地提高效率；
- C 语言的注释有两种/* 和/* 括起来的文字，可以跨行，都被视为注释内容；
- 也可以用简单方便的//来注释当前行从//开始的文字；
- 注释后的文字会被编译器忽略掉，意味着注释的文字不会对程序的功能或执行产生任何影响。

第七个组成部分

注释：/*.....*/和//

- 注释就是在程序源代码文件中添加说明性文字，方便记忆、阅读；
- 程序代码写了后，时间长了，就会忘记其含义，当再次阅读或别人阅读，要进行修改时，注释会极大地提高效率；
- C 语言的注释有两种/* 和/* 括起来的文字，可以跨行，都被视为注释内容；
- 也可以用简单方便的//来注释当前行从//开始的文字；
- 注释后的文字会被编译器忽略掉，意味着注释的文字不会对程序的功能或执行产生任何影响。

第七个组成部分

注释：/*.....*/和//

- 注释就是在程序源代码文件中添加说明性文字，方便记忆、阅读；
- 程序代码写了后，时间长了，就会忘记其含义，当再次阅读或别人阅读，要进行修改时，注释会极大地提高效率；
- C 语言的注释有两种/* 和/* 括起来的文字，可以跨行，都被视为注释内容；
- 也可以用简单方便的//来注释当前行从//开始的文字；
- 注释后的文字会被编译器忽略掉，意味着注释的文字不会对程序的功能或执行产生任何影响。

第七个组成部分

注释：/*.....*/和//

- 注释就是在程序源代码文件中添加说明性文字，方便记忆、阅读；
- 程序代码写了后，时间长了，就会忘记其含义，当再次阅读或别人阅读，要进行修改时，注释会极大地提高效率；
- C 语言的注释有两种/* 和/* 括起来的文字，可以跨行，都被视为注释内容；
- 也可以用简单方便的//来注释当前行从//开始的文字；
- 注释后的文字会被编译器忽略掉，意味着注释的文字不会对程序的功能或执行产生任何影响。

第八个组成部分

花括号：{}

- C 语言用花括号{}来表示“块”，用于限定范围：如main()函数定义中函数体的具体内容就用花括号包括起来，形成一个整体“块”；
- 可以将花括号内所有的语句看成一个整体，一个“复合”语句，一个有很多语句合成的语句。

第八个组成部分

花括号：{}

- C 语言用花括号{}来表示“块”，用于限定范围：如main()函数定义中函数体的具体内容就用花括号包括起来，形成一个整体“块”；
- 可以将花括号内所有的语句看成一个整体，一个“复合”语句，一个有很多语句合成的语句。

Outline

C 语言组成练习

寻找指定的部分

```
1  #include <stdio.h>
2   void display_line(void);
3   main(){
4   display_line();
5   printf("\n␣Teach␣Yourself␣C␣In␣21␣
        Days!\n");
6   display_line();
7   return 0;
8   }
9   /* print asterisk line */
10  void display_line(void){
11  int counter;
12  for( counter = 0; counter < 21;
        counter++ )
13  printf("*" );      }
```

C 语言组成联系

寻找指定的部分

- 哪些是执行语句？
- 哪些语句定义了变量？
- 哪些语句是函数原型说明？
- 哪些语句包括了函数定义？
- 哪些是注释？

C 语言组成练习

读程序：程序完成什么任务？

```
1  #include <stdio.h>
2  main(){
3  int ctr;
4  for( ctr = 65; ctr < 91; ctr++ )
5  printf("%c", ctr );
6  return 0;
7  }
8  /* end of program */
```

输入、编译、执行、查看运行结果。

C 语言组成练习

读程序：程序完成什么任务？

```
1  #include <stdio.h>
2   #include <string.h>
3   main()
4   {
5   char buffer[256];
6   printf( "Enter␣your␣name␣and␣press␣<
        Enter>:\n");
7   gets( buffer );
8   printf( "\nYour␣name␣has␣%d␣
        characters␣and␣spaces!",
9   strlen( buffer ));
10  return 0;
11  }
```

输入、编译、执行、查看运行结果。