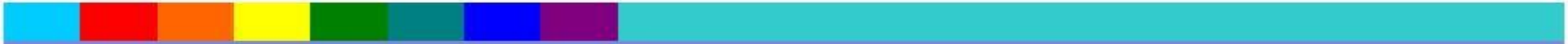
南京大学 计算机科学与技术系
Department of Computer Science & Technology, NJU

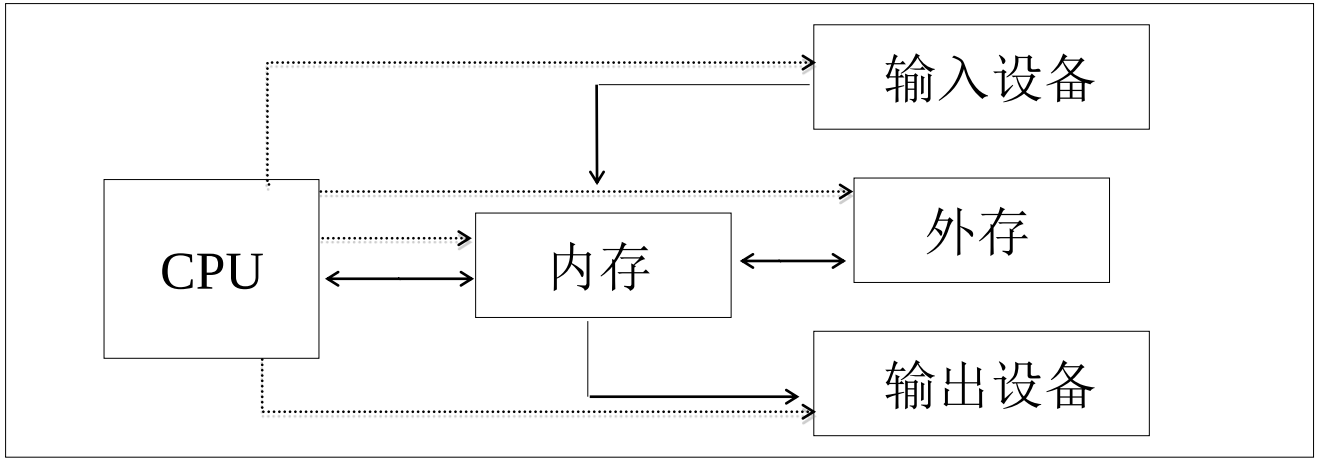
the first step

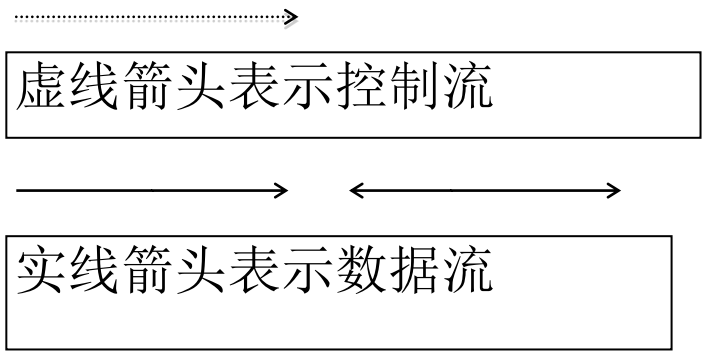
硬件、软件与程序



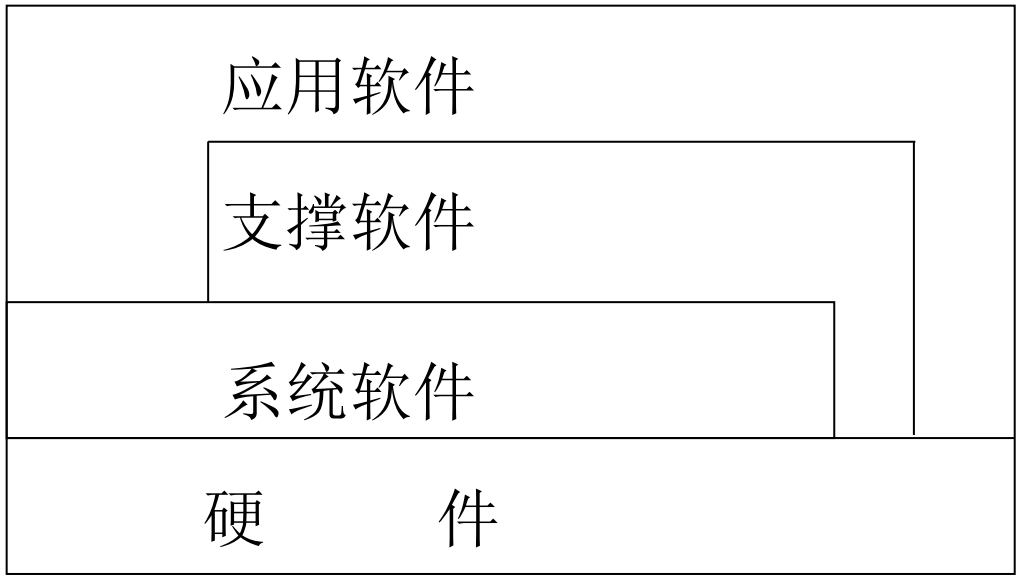
刘奇志

- **硬件（hardware）指的是组成计算机的元器件和设备**
  - 中央处理器（CPU，Central Process Unit）
  - 内存（memory）
  - 外围设备
    - 外存（external storage，例如硬盘、U盘、光盘等）
    - 输入设备（例如键盘、鼠标等）
    - 输出设备（例如显示器、打印机等）





- **软件（software）指的是计算机系统中的程序及相关文档。**
  - 系统软件（例如Windows、Unix、Linux、Mac OS等）
  - 支撑软件（例如集成开发环境、软件测试工具等）
  - 应用软件（例如财务软件、自动控制软件等）



程序（program）

- **一组连续的相互关联的计算机指令**
  - CPU能执行的指令包括：
    - 算术运算指令：实现加、减、乘、除等；
    - 比较指令：比较两个操作数的大小；
    - 数据传输指令：实现CPU的寄存器、内存以及外设之间的数据传输；
    - 流程控制指令：用于确定下一条指令的内存地址
      - ✓ 顺序
      - ✓ 转移
      - ✓ 循环
      - ✓ 子程序调用/返回

- **指示计算机处理某项计算任务的任务书，计算机根据该任务书，执行一系列操作，并产生有效的结果。**
  - **计算（compute）：并非单指 数值计算，而是指根据已知数据（data）经过一定的步骤（算法）获得结果的过程**

语言（language）

- 现在的计算机还不能很好地理解人类的自然语言，所以一般不能用自然语言直接进行程序设计（programming）。
- 研究人员已经发明了多种程序设计语言，以便程序员设计程序。
- 利用程序设计语言设计、编写的程序（源程序），通过相应的翻译、优化工具，可以形成计算机能够理解的机器语言程序（目标程序）。
  - 机器语言只有0、1两种符号
  - 目标程序经处理可以被计算机执行

南京大学 计算机科学与技术系
Department of Computer Science & Technology, NJU

Chapter 0

初识C程序

刘奇志

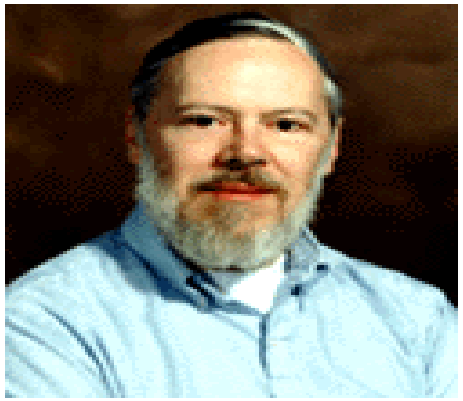
- **C语言简介**
- **C程序基本结构与main函数**
- C语言的字符集/单词/符号常量
- C语言的操作符和表达式
- C语言的标点符号与注释
- C语言的语句
- **C语言常用的简单数据类型**
- **变量（定义、赋值与初始化、值的输入）**
- **数据的输出**

C语言的来历

- **ALGOL 60 (algorithmic language，国际委员会，1960)**
  - 简洁、科学的定义
- **CPL (combined programming language，剑桥、伦敦大学，1963)**
  - 接近硬件、规模大
- **BCPL (basic ~，剑桥大学Matin Richards，1967)**
  - 简化
- **B (贝尔实验室Ken Thompson，1970)**
  - 精华
- **C (贝尔实验室D.M.Ritchie，1972~1973)**
  - 既保持了BCPL和B语言的优点(精练、高效、接近硬件等)
  - 又克服了它们的缺点(过于简单、数据无类型、功能有限等)

- **C++ (贝尔实验室Bjarne Stroustrup，1979)**
  - 为支持面向对象程序设计而设计(先是C with Class)

C语言之父



- **Dennis M. Ritchie(1941-2011)**
  - 1967起一直在位于美国新泽西州的贝尔实验室工作
  - 他的工作得到了很多计算机组织的公认和表彰
    - 美国计算机协会(ACM)授予的系统及语言杰出论文奖(1974)
    - 电气和电子工程师协会(IEEE)的先驱奖(Emmanuel Piore) (1982)
    - 图灵奖(Turing) (1983)
    - …

ACM: **A**ssociation for **C**omputing **M**achinery

IEEE: **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers

C语言标准（specification）

- 1978年Dennis M. Ritchie与Brian W.Kernighian出版了《 The C Programming Language》，此书是最初的C语言标准（K&R C）
- 随着C语言的应用和发展，形成了多种C语言的实现（**implementation**）版本，各种版本在功能和函数库的设置内容上存在差别。1983年， ANSI开始制定统一的C语言标准，直至1989 年底正式批准名为ANSI X3.159-1989的标准（C89），1990年，ISO采纳了C89并以ISO/IEC 9899:1990颁布
- 2000年初，ISO颁布了ISO/IEC 9899:1999（**C99**）
- 2011年底，ISO发布了ISO/IEC 9899:2011 （C11）

- Stroustrup一直积极推动C++语言的标准化，1998年底，ANSI/ISO 发布了ISO/IEC 14882:1998（**C++98**）
- 2011年底， ISO发布了ISO/IEC 14882:2011 （C++11）

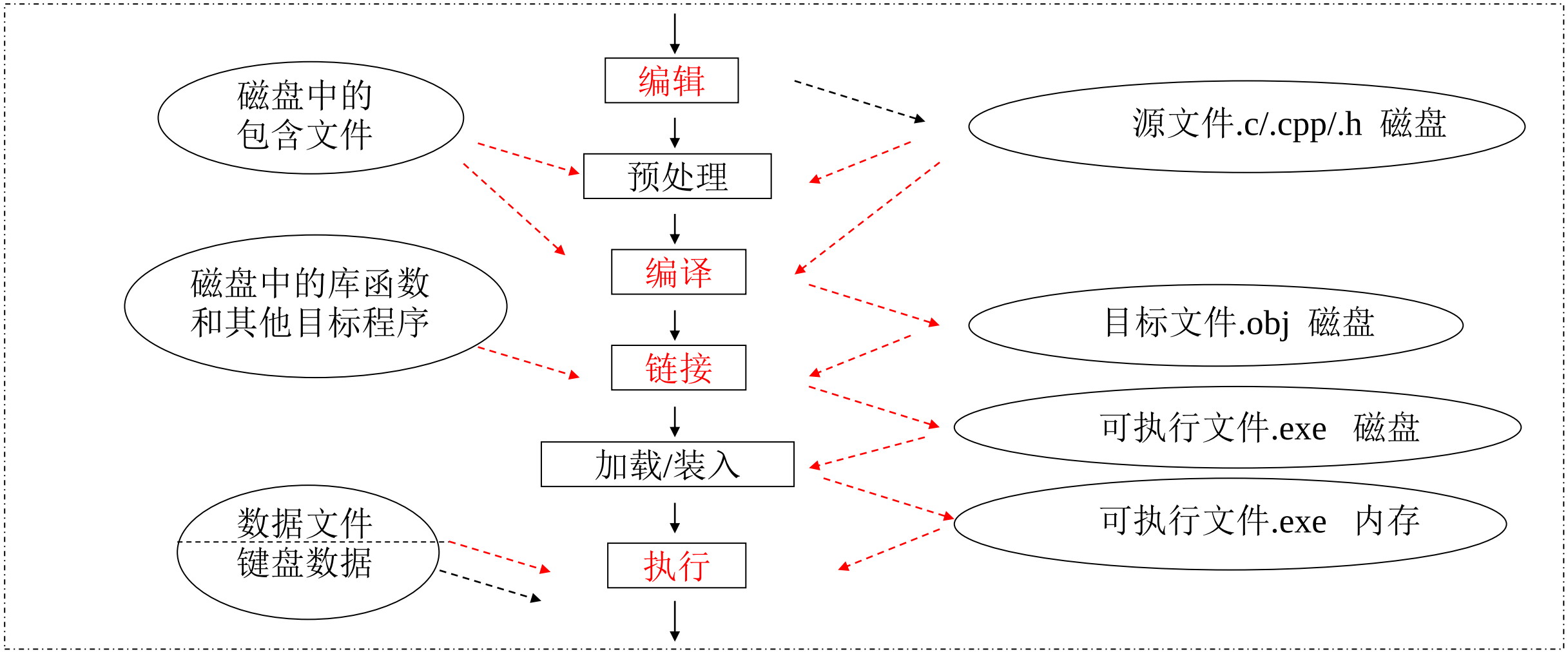
ANSI: American National Standards Institute

ISO: International Organization for Standardization

C程序的运行步骤

- **编辑(.c/.cpp/.h) → 编译(.obj) → 链接(.exe) → 执行 → 输出结果**

edit pre-process compile link load execute/Run output



C程序开发环境

- 上述C程序的编辑、编译等步骤都需要特定软件的支撑。
- 通常使用带有编辑程序、编译程序等支撑软件的集成开发环境（IDE）进行程序开发。
  - Visual Studio C++
  - Dev-C++
- 程序员必须合理设计并按照规定来编辑程序，不合规定或逻辑错误的程序不能正常编译、链接或执行，IDE通常含有帮助程序员设计和调试（debug）程序的软件。
- 操作过程中，有可能发现程序有错，需要修改程序，整个过程可能会重复多次，直到得出正确的执行结果。在这些集成环境中，往往使用一条命令（菜单）就能完成所有的步骤。

- 不同的IDE所含的编译程序及其对标准的支持程度不尽相同
- 程序开发的几个步骤在不同开发环境中的安排也不尽相同
  - Visual Studio C++中的Build菜单包含编译和链接步骤，执行步骤一般安排在Debug菜单中
  - Dev-C++中，编译、链接和执行步骤则安排在Execute菜单下

- 初学者应该以掌握程序设计的基本方法为目标，尽量避免被语言标准及各个开发环境实现的细节所纠缠，有些问题和概念在计算机系统原理、编译技术、程序设计语言概论等后续课程中讨论更为合适。

C程序的基本结构与main函数

- **一个C程序可以由若干个变量与函数组成。变量是操作的对象与结果，函数是计算的过程，含若干条语句**
- **一个C程序必须定义一个名字为的main函数。**
  - 一个简单的C程序只定义一个main函数；
  - 一个复杂的C程序最好定义多个函数，但其中main函数必须有且只有一个；
  - 可以把main函数的代码保存在文件main.c中。

- 每一个程序都应该有的内容（灰色内容可以省略）：

```
int main(void)
{


    return 0;
}
```

该函数没有任何实质性的计算，它构成一个可执行的空程序。

- int是约定的main函数返回值的类型，表示main函数执行完毕返回给执行环境的信号是一个整数，与return后面的0一致

- 函数名后面必须有一对圆括号，里面的void（空类型）表示该main函数不带参数，void可以省略

- 接下来必须是一对花括号，花括号里是函数体
- 一般约定正常程序段的末尾写“return 0;”，异常处理程序段（实际应用中往往要编写专门的异常处理程序段）的末尾写“return -1;”。执行环境根据接收到的-1或0，决定要不要采取故障恢复等措施

- 执行程序时，总是从main函数体中的第一条语句开始执行，当执行到main函数中的return语句时，整个程序结束

```c
#include <stdio.h>

int main()
{     printf("Now Join Us! \n");
      return 0;
}
```

该程序增加了一条执行输出功能的语句，main函数上面的内容辅助完成输出功能。

```c
#include <stdio.h>
int main()
{
    printf("Now Join Us! \n");
    return 0;
}
```

```c
#include <stdio.h>
main()
{
    printf("Now Join Us! \n");
}
```

不好的习惯

```c
#include <stdio.h>
void main()
{
    printf("Now Join Us! \n");
}
```

不好的习惯

- C语言简介
- C程序基本结构与main函数
- **C语言的字符集/单词/符号常量**
- **C语言的操作符和表达式**
- **C语言的标点符号与注释**
- **C语言的语句**
- **C语言常用的简单数据类型**
- **变量（定义、赋值与初始化、值的输入）**
- **数据的输出**

关键字
标识符
字面常量

C语言的字符集（symbol set）

- **构成语言的基本符号**

- **C语言的字符集**
  - 大小写英文字母
  - 阿拉伯数字
  - 特殊符号

    ~ ！ # % ^ & * _ – + = | \ : ; " ' , . ? / ( ) { } [ ] < > 制表 空格 回车换行

- **程序中不能出现字符集之外的字符（双引号里除外）。**

× √ π ` @ $ “” , ‘’ ;

×

键盘上没有的符号

键盘上有的符号

汉字库里的符号

比如：

```
#include <stdio.h>

int main()
{     printf("Now Join Us! \n");
      return 0;
}
```

√

没有出现字符集之外的字符

```
#include <stdio.h>

int main()
{     printf( “Now Join Us! \n” );
      return 0;
}
```

×

出现了字符集之外的字符

一个更为完整的例子

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
    while(d <= n)
    {
        sum = sum + PI * d;
        d = d + 1;
    }

    printf("The sum: %f", sum);
    printf("%c", mu);    //显示计量单位
```

C语言的单词（token）

- **由字符集中的字符按照一定规则构成，语言的基本单位**
- **包括：**
  - 关键字
  - **标识符**
  - 字面常量

  - 单词与单词之间用空格分隔

关键字（keyword）

- 保留词汇，有固定的作用和含义，通常由小写字母组成，在程序中不能用作其他目的。

- 表示语句：break、continue、do、else、for、goto、if、return、switch、while…
- 表示数据类型：auto、char、const、double、enum、float、int、long、register、short、signed、struct、union、unsigned、void…
- 表示标号：case、default…
- 其他关键字：extern、sizeof、static、typedef…

关键字（keyword）

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
    while(d <= n)
    {
        sum = sum + PI * d;
        d = d + 1;
    }

    printf("The sum: %f", sum);
    printf("%c", mu);    //显示计量单位
```

标识符（identifier）

- **由字符集中的大小写英文字母、阿拉伯数字和下划线组成，且首字符是字母或下划线**
  - i、price_car、myFun、Node_3、Loop4、PI、pi

    5x、stu.score、number 1、y-average ✗

- **可以用作符号常量、变量、函数、形式参数、类型、以及语句标号等的名称。**

- **程序中的标识符必须有定义（definition），即必须赋予某标识符一定的含义，没有定义的（undefined）标识符不能使用**
- **系统预定义标识符**
  - 在源代码中前面加#，编译后被相应的内容取代，如：define、include、if、ifdef、else、endif、...
  - 其他预定义标识符，如：main、printf…
- **自定义标识符**
  - 不能和关键字或预定义标识符重复。 `int` ×
  - 在相同的有效范围内，已定义的标识符不能重复定义（redeclaration）
  - 不同的标识符，其定义的具体格式不同

- **此外，还需要注意标识符的命名风格**
  - 尽量不与其他标识符同名。
  - 尽量不在同一个程序中定义拼写相同、大小写不同的标识符。
    - 标识符中的大写字母与小写字母是有区别的，比如`PI`和`pi`是两个不同的标识符。
  - 尽量使用不太长的有意义的单词
    - 标识符的有效长度由具体编译器决定。

预定义标识符

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
      while(d <= n)
      {
            sum = sum + PI * d;
            d = d + 1;
      }

      printf("The sum: %f", sum);
      printf("%c", mu);    //显示计量单位
```

自定义标识符

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
        while(d <= n)
        {
              sum = sum + PI * d;
              d = d + 1;
        }

        printf("The sum: %f", sum);
        printf("%c", mu);    //显示计量单位
```

字面常量（literal constant）

- 常量用于表示在程序执行过程中不会改变或不允许被改变的数据，如：闰年的天数、圆周率等。

- 程序中直接书写的常量
  - 整数（如7等）
  - 小数（如3.14）
  - 字符常量（如'm'）
  - 字符串常量（如"Hello World!"）

字面常量（literal constant）

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
        while(d <= n)
        {
              sum = sum + PI * d;
              d = d + 1;
        }

        printf("The sum: %f", sum);
        printf("%c", mu);    //显示计量单位
```

C语言的操作符(operator)与表达式(expression)

- **操作符，用于描述程序中的操作，又叫运算符**
  - 系统定义了一批操作符的功能和操作规则，以实现基本运算
    - 如=、<=、+、*、 ()、-、/、%　　求余数运算
  - 字面常量、符号常量、变量与函数可以作为操作符的基本操作对象，又叫操作数（operand）
- **表达式是用操作符将操作数连接起来的式子**
  - 如d = d + 1
  - 表达式也可以作为操作数参加运算，如 d + 1
  - 最简单的表达式是一个操作数，如一个字面常量1，一个变量d
  - 书写的时候，操作符两端各加一个空格一般可以提高可读性

操作符(operator)

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
      while(d <= n)
      {
            sum = sum + PI * d;
            d = d + 1;
      }

      printf("The sum: %f", sum);
      printf("%c", mu);     //显示计量单位
```

C语言的标点符号(punctuation)与注释(comment)

- **标点符号在程序中起到某些语法、语义上的作用，特别是分隔作用。**
  - 井号（#）表示预处理命令行
  - 分号（;）可以表示一条语句的结束
  - 逗号（,）可以分隔多个参数或变量
  - 空格（ ）可以作为词汇之间的间隔
  - 冒号（:）可以分隔语句标号与语句
  - 行尾的反斜杠（\）是续行符，表示行尾的**单词**未完待续
  - . . .
  - 标点符号与相邻的单词之间一般不必加空格，分号、逗号、冒号后加一个空格可提高可读性
- **注释不被编译和执行，用来提示或解释程序的含义，在调试程序时，对暂时不执行的语句也可用注释符分离出来。**
  - 多行注释：以/*开始，以*/结束
  - 单行注释：以//开始

标点符号(punctuation)

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
        while(d <= n)
        {
              sum = sum + PI * d;
              d = d + 1;
        }

        printf("The sum: %f", sum);
        printf("%c", mu);     //显示计量单位
```

C语言的语句（statement）

- **表达式末尾加一个分号可以构成语句**
  - `d = d + 1;`
- **最简单的语句是一个分号，即空语句，不执行任何操作**
- **可以用一对花括号将多个语句括起来，形成复合语句，一个复合语句可以看作一个语句块**
  - ```
    {
      sum = sum + PI * d;
      d = d + 1;
    }
    ```
- **一些关键字，如while、return等，与表达式、分号或（子）语句配合也可以构成语句**
    ```

语句

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
while(d <= n)
{
      sum = sum + PI * d;
      d = d + 1;
}

printf("The sum: %f", sum);
printf("%c", mu);     //显示计量单位
```

C语言的符号常量（manifest constant)

- 符号常量（又叫命名常量）：对于程序中多次使用的字面常量，可以定义成符号常量，即通过常量定义给常量取一个名字，在程序中通过常量名来使用这个字面常量。
- 符号常量是标识符的一种

常量定义

- **#define <符号常量> <字面常量>**
  - **如 #define PI 3.1415926**
  - 这里，系统预定义标识符define 是预处理命令，预处理器会在编译前把源程序语句中的符号常量全部替换成相应的字面常量，参加编译的是替换后的字面常量。
  - 预处理命令行的末尾一般不加分号，常量定义的末尾如果有分号，分号会被当成是字面常量的一部分，从而有可能造成语法错误。
  - 这种定义方式又叫宏（Macro）定义，不会像变量定义那样检查类型。

- **常量定义的好处**
  - 增加程序的**易读性**
  - 保证程序对常量使用的**一致性**
  - 增强程序的**易维护性**

符号常量

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
      while(d <= n)
      {
            sum = sum + PI * d;
            d = d + 1;
      }

      printf("The sum: %f", sum);
      printf("%c", mu);    //显示计量单位
```

用符号常量（少循环一次）

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
        while(d <= n)
        {
              sum = sum + PI * d;
              d = d + 1;
        }

        printf("The sum: %f", sum);
        printf("%c", mu);    //显示计量单位
```

- C语言简介
- C程序基本结构与main函数
- C语言的字符集/单词/符号常量
- C语言的操作符和表达式
- C语言的标点符号与注释
- C语言的语句
- **C语言常用的简单数据类型**
- **变量（定义、赋值与初始化、值的输入）**
- **数据的输出**

C语言常用的简单数据类型

- 程序的任务通常与数据处理有关
- 数据被分成若干种不同的类型
- 常用的基本类型
  - 整型 `int`
  - 实型 `double`
  - 字符型 `char`
  - 空类型 `void`

C语言变量

- **变量的定义（definition）**
  - 通过列出变量的类型及其名字来规定变量的类型和名字属性
  - 相同类型的多个变量可以并列定义
  - 可以在程序中随时定义变量

例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

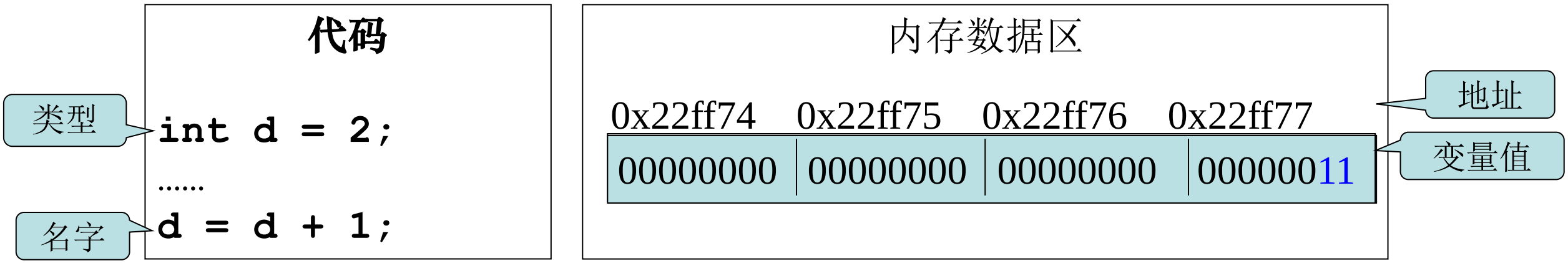
```
        while(d <= n)
        {
              sum = sum + PI * d;
              d = d + 1;
        }

        printf("The sum: %f", sum);
        printf("%c", mu);    //显示计量单位
```

变量的属性

- 程序执行到变量定义处，即意味着系统要为变量分配一定大小的空间，以准备存储变量的值。
- 存储空间里起初是一些0/1组成的无意义的值，可以通过赋值或输入值来获得有意义的值。
- 存储空间由地址来标识，一般由系统自动管理。

- 可见，变量具有程序中可见的类型和名字属性，还具有程序中不一定可见的值属性，以及程序中一般不可见的内存地址属性。



变量的赋值（assignment）与初始化（initialize ）

- 通过给变量赋值，可以使变量获得有意义的值，还可以使变量的值在程序运行过程中发生改变。
- C语言中用一个等于号表示赋值，这里的等于号可以理解为←，即将右边的值存入左边变量的内存中，会改写左边变量的内存里原来的值。
- 变量的值可以在定义的时候赋一个初值，即初始化（在程序编译期间完成赋值任务），如：

```
int n, d = 2;
```

C++11新标准：
列表初始化：`int i = {0}; int j{0};`

- 也可以后期赋值（在程序执行期间完成赋值任务），如：

```
d = d + 1;
```

- 如果定义的变量未初始化（uninitialized），则不确定的值没有被替换，可能会给后面的相关计算带来预想不到的结果。
- 尽可能在定义变量的同时初始化该变量，如果变量的引用处和其定义处相隔比较远，变量的初始化很容易被忘记。

例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
while(d <= n)
{
    sum = sum + PI * d;
    d = d + 1;
}

printf("The sum: %f", sum);
printf("%c", mu);    //显示计量单位
```

变量mu可以省略，直接输出表示计量单位的字符m

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;        //d初始值是2
  double sum = PI;     //sum初始值是直径为1的圆的周长

  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
    while(d <= n)
    {
        sum = sum + PI * d;
        d = d + 1;
    }

    printf("The sum: %fm", sum);
```

变量值的输入

- 输入（input），一般是指在程序执行过程中，将需要的数据从键盘输入内存
- 程序员可以在标准函数库的基础上实现输入功能
- C语言调用函数库中的scanf()等函数，需要在程序头部用#include <stdio.h>包含输入库函数的说明信息

- VS: scanf_s

变量值的输入

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

格式符

```
      while(d <= n)
      {
            sum = sum + PI * d;
            d = d + 1;
      }

      printf("The sum: %f", sum);
      printf("%c", mu);     //显示计量单位
```

C语言程序中数据的输出

- 输出（output），一般是指将程序执行的结果显示到显示器上
- 程序员可以在标准函数库的基础上实现输出功能
- C语言调用函数库中的printf等输出函数，需要在程序头部用#include <stdio.h> 包含输出函数的说明信息

C语言程序中数据的输出

- 例0.1 计算一组圆（直径为n以内的正整数）的周长之和（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;          //d初始值是2
  double sum = PI;       //sum初始值是直径为1的圆的周长
  char mu = 'm';
  printf("Input n: ");
  scanf("%d", &n);
  ......
  return 0;
}
```

```
    while(d <= n)
    {
        sum = sum + PI * d;
        d = d + 1;
    }

    printf("The sum: %f", sum);
    printf("%c", mu);     //显示计量单位
```

C++兼容的C程序

C++语言兼容C语言的输入输出方式，并将stdio.h中的内容包含在cstdio中，即程序头部写成#include <cstdio>

```
#include <stdio.h>
#define PI 3.14
int main( )
{     ......
      printf("Input n: ");
      scanf("%d", &n);
      ......
      return 0;
}
```

```
#include <cstdio>
#define PI 3.14
int main( )
{     ......
      printf("Input n: ");
      scanf("%d", &n);
      ......
      return 0;
}
```

C++程序

C++语言调用类库中的输入对象cin/输出对象cout，需要在程序头部用#include <iostream> 包含输入/输出对象的说明信息，并用using namespace 指明输入/输出对象名是在标准名空间std中定义的。

```
#include <iostream>
using namespace std;
#define PI 3.14
int main( )
{     ……
      cout << "Input n: ";
      cin >> n;
      ......
      cout << "The sum is: " << sum << mu;
      return 0;
}
```

C程序

```c
#include <stdio.h>
int main()
{     printf("Now Join Us! \n");
      return 0;
}
```

C++兼容的C程序

```c
#include <cstdio>
int main()
{     printf("Now Join Us! \n");
      return 0;
}
```

C++程序

```c
#include <iostream>
using namespace std;
int main()
{     cout << "Now Join Us! " << endl;
      return 0;
}
```

补充几个输入输出例子

```c
#include <stdio.h>
int main()
{
	char ch;
	ch = getchar();
	return 0;
}//C
```

```cpp
#include <iostream>
using namespace std;
int main()
{
	char ch;
	cin >> ch;
	return 0;
}//C++
```

```c
#include <stdio.h>
int main()
{
      int m, n;
      scanf("%d%d", &m, &n);
      printf("%d\n", m + n);
      printf("%d\n", m * n);
      return 0;
}//C
```

```
567 85
652
48195
```

```
567
85
652
48195
```

转义符 (escape sequence)

\n	回车换行
\"	显示双引号本身
\%	显示百分号本身
\\	显示反斜杠本身

```
#include <iostream>
using namespace std;
int main()
{
      int m, n;
      cin >> m >> n;
      cout << m + n << endl;
      cout << m * n << endl;
      return 0;
}//C++
```

回车换行

```
567 85
652
48195
```

```
567
85
652
48195
```

输入后面不能加endl，输出可以加；
输入用>>，输出用<<，莫搞反了！！

如果没有回车换行呢？Try!

```c
#include <stdio.h>
int main()
{
	int m, n;
	scanf("%d%d", &m, &n);
	printf("%d + %d = %d \n", m, n, m + n);
	printf("%d * %d = %d \n", m, n, m * n);
	return 0;
}//C
```

```
567
85
567 + 85 = 652
567 * 85 = 48195
```

```cpp
#include <iostream>
using namespace std;
int main()
{
	int m, n;
	cin >> m >> n;
	cout << m << " + " << n << " = " << m + n << endl;
	cout << m << " * " << n << " = " << m * n << endl;
	return 0;
}//C++
```

```
567
85
567 + 85 = 652
567 * 85 = 48195
```

```c
#include <stdio.h>
int main()
{
	double m, n;
	scanf("%lf%lf", &m, &n);
	printf("%f\n", m + n);
	return 0;
}//C
```

```
567.1  85.1
652.2
```

```
567.1
85.1
652.2
```

```
#include <iostream>
using namespace std;
int main()
{
      double m, n;
      cin >> m >> n;
      cout << m + n << endl;
      return 0;
}//C++
```

```
567.1 85.1
652.2
```

```
567.1
85.1
652.2
```

```c
#include <stdio.h>
int main()
{     printf("a simple calculation: \n");
      printf("567 + 85 = %d \n", 567 + 85);
      printf("567 * 85 = %d \n", 567 * 85);
      printf("5.67 * 8.5 = %.1f\n", 5.67 * 8.5);
      printf("%c\n", 'A');
      return 0;
}//C
```

```
a simple calculation:
567 + 85 = 652
567 * 85 = 48195
5.67 * 8.5 = 48.2
A
```

```
a simple calculation:
567 + 85 = 652
567 * 85 = 48195
5.67 * 8.5 = 48.2
A
```

```
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
      cout << "a simple calculation: " << endl;
      cout << "567 + 85 = " << 567 + 85 << endl;
      cout << "567 * 85 = " << 567 * 85 << endl;
      cout << "5.67 * 8.5 = " << fixed << setprecision(1)
           << 5.67 * 8.5 << endl;
      cout << 'A' << endl;
      return 0;
}//C++
```

良好的编程习惯

好的程序：

正确（correct）
可靠（reliable）
高效（efficient）
易读（readability）
可重用（re-usable）
可移植（portable）
……

- 确保**正确**的算法、数据结构与代码
- 采用**适合计算机的算法**
- **合理**组织数据
- **提高程序的易读性**
  - 注意自定义标识符的命名风格
  - 注意程序的排版
  - 为程序书写注释，注释的位置应与被描述的代码相邻，可以放在代码的上方或右方；当代码比较长，特别是有多重嵌套时，应在一些段落的结束处加注释
  - …

C程序的书写比较自由，不必在规定的行或列 书写规定的内容。

不过良好的书写格式不仅可以使程序美观，还有利于提高程序的可读性，便于程序的调试和维护。

初学者应注意养成良好的书写习惯，

- C程序的书写比较自由，不必在规定的行或列 书写规定的内容。
- 不过良好的书写格式不仅可以使程序美观，还有利于提高程序的可读性，便于程序的调试和维护。
- 初学者应注意养成良好的书写习惯，比如：
  - 一行只写一个语句
  - 采用好的缩进模式（即在同一块语句前插入等量的空格-用Tab键，并保持前后一致）
  - 在运算符与操作数之间恰当地添加空格
  - 在程序段落之间恰当地添加空行
  - ...

```
#include<stdio.h>
#define PI 3.14
int main( )
{int n,d=2;double sum=PI;
printf("Input n: ");
scanf("%d",&n);
while(d<=n)
{sum=sum+PI*d;d=d+1;
}
printf("The sum:%fm",sum);
return 0;
}
```

```
#include<stdio.h>
#define PI 3.14
int main( )
{int n,d=2;
double sum=PI;
printf("Input n: ");
scanf("%d",&n);
while(d<=n)
{sum=sum+PI*d;
d=d+1;
}
printf("The sum:%fm",sum);
return 0;
}
```

一行只写一句!

```
#include<stdio.h>
#define PI 3.14
int main( )
{ int n,d=2;
  double sum=PI;
  printf("Input n:");
  scanf("%d",&n);
  while(d<=n)
  {  sum=sum+PI*d;
     d=d+1;
  }
  printf("The sum:%fm",sum);
  return 0;
}
```

缩进！

```
#include <stdio.h>
#define PI 3.14
int main( )
{ int n, d = 2;
  double sum = PI;
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  {  sum = sum + PI * d;
     d = d + 1;
  }
  printf("The sum: %fm", sum);
  return 0;
}
```

操作符两端加空格！

```
#include <stdio.h>
#define PI 3.14

int main( )
{
  int n, d = 2;
  double sum = PI;
  printf("Input n: ");
  scanf("%d", &n);

  while(d <= n)
  {
     sum = sum + PI * d;
     d = d + 1;
  }
```

适当空行！

```c
#include <stdio.h>
#define PI 3.14

int main( )
{
  int n, d = 2;
  double sum = PI;
  printf("Input n: ");
  scanf("%d", &n);

  while(d <= n) {
     sum = sum + PI * d;
     d = d + 1;
  }

  printf("The sum: %fm", sum);
```

花括号问题

关于自定义标识符命名规则

- 自定义标识符命名在项目中往往是一个比较难以处理的议题，程序员倾向于使用其个人的命名约定，而不喜欢别人规定他们如何编写代码。

- 然而，当代码需要被团队内的其他成员阅读时(特别是代码检查的时候)，拥有通用的命名约定是很有价值的，也便于 自己日后再阅读自己的代码。

- 一直以来,最流行的变量命名约定是所谓的匈牙利表示法(Hungarian Notation)，最初由Microsoft的Charles Simonyi提出，并且在Microsoft内部使用了许多年。

  (这个约定规定了以标准的3或4个字母前缀来表示变量的数据类型，比如表示学生年龄的整型变量就应该命名为intStuAge.)

本课程自定义标识符命名具体建议☆

- 【总则】采用一致的、有意义的标识符名字。对不同种类的标识符最好采用不同风格的名字。

- 【建议……】
  - 占分！

- **本课程课件有时没有遵循所建议的规则，这是为了将相关内容放在一张幻灯片上，便于讲解。**

本课程自定义标识符命名具体建议☆

- 【总则】采用一致的、有意义的标识符名字。对不同种类的标识符最好采用不同风格的名字。

- 【建议1】自定义标识符应当直观，用词尽量准确，可望文知意。切忌使用**汉语拼音简拼**来命名。

- 【建议2】标识符的长度应当符合“min-length && max-information”原则。一般来说，长名字能更好地表达含义，但名字并非越长越好，单字符的名字也是有用的，常见的如i，j，k，m，n，x，y，z等，它们通常可用作函数内的局部变量。

- 【建议3】程序中不要出现仅靠大小写区分的相似的标识符。例如：

```
int   x, y, X;    // 变量x 与 X 容易混淆
void foo(int x);     // 函数foo 与FOO容易混淆
void FOO(int y);
```

- 【建议4】用一对反义词命名具有相反含义的变量或函数等。例如：

```
int     minValue, maxValue;
int     SetValue(…), GetValue(…);
```

- 【建议5】 函数名和类型名用大写字母开头的单词组合而成。如：

```
void  Init(void);
void  SetValue(int value);
```

系统定义的类型名、main函数名及库函数名除外

- 【建议6】 变量名和参数名的首单词用小写字母开头。如：

```
int flag;
int stuAge;
int current_value
```

- 【建议7】习惯使用符号常量，符号常量名全用大写字母，用下划线分割单词。如：

```
#define MAX_LENGTH  100
#define PI 3.14
```

小结

C程序的组成

C语言基本元素
字符集
单词
表达式
语句
语句块
函数
模块
程序

- C的**单词**
  - 关键字
  - 标识符
    - 预定义标识符
    - 自定义标识符
  - 字面常量
- 操作符
- 标点符号

符号常量

变量

C语言常用的简单数据类型
变量（定义、赋值与初始化、值的输入）
数据的输出

变量的属性

- 要求：
  - 会编写简单的C程序
    - 一个程序代码量≈10行，
      在main函数中完成变量定义、输入、简单处理、输出
  - 熟练C语言程序的上机步骤
  - C语言的基本词法
  - 按建议规则编程，养成良好的编程习惯
- 作业：见课程网站

Thanks !