

南京大学 计算机科学与技术系

Department of Computer Science & Technology, NJU

the first step

认知与体验（硬件、软件与C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)

- 运算器 (ALU, Arithmetic Logic Unit)
- 控制器 (CU, Control Unit)
- 寄存器 (registers)

◆ 内存 (primary storage / internal memory/main memory)

◆ 外围设备

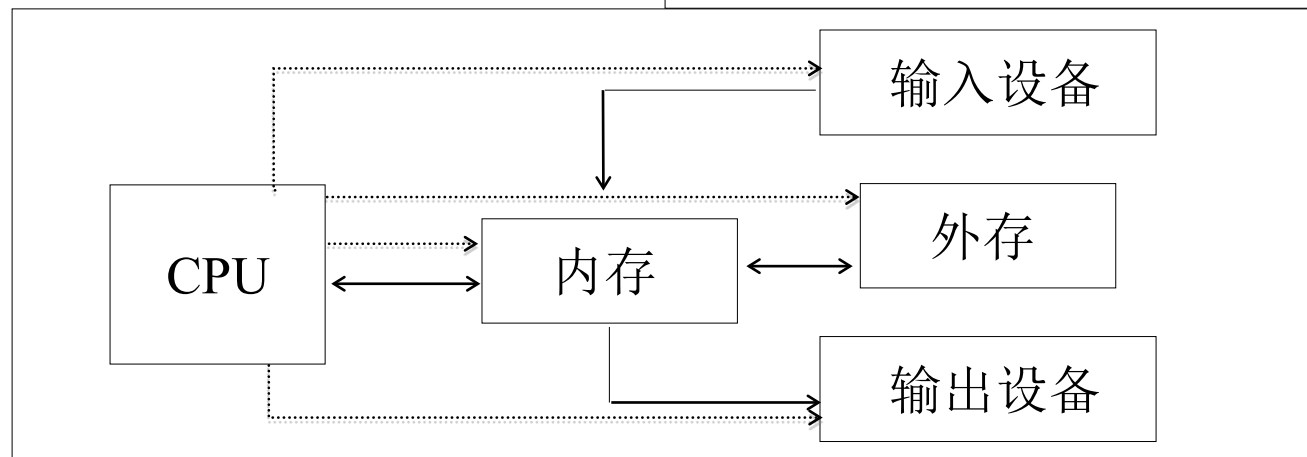
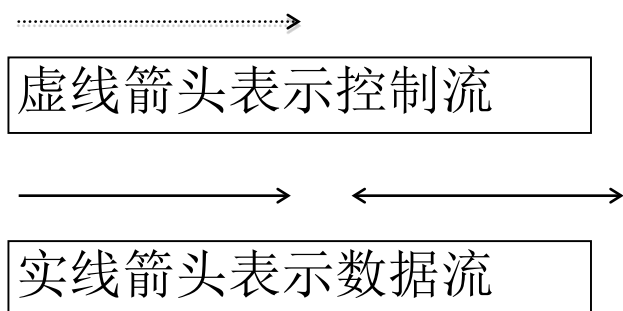
- 外存 (secondary storage/auxiliary storage / external memory)
- 输入设备 (input devices)
- 输出设备 (output devices)

存储器 (storages)

位: bit

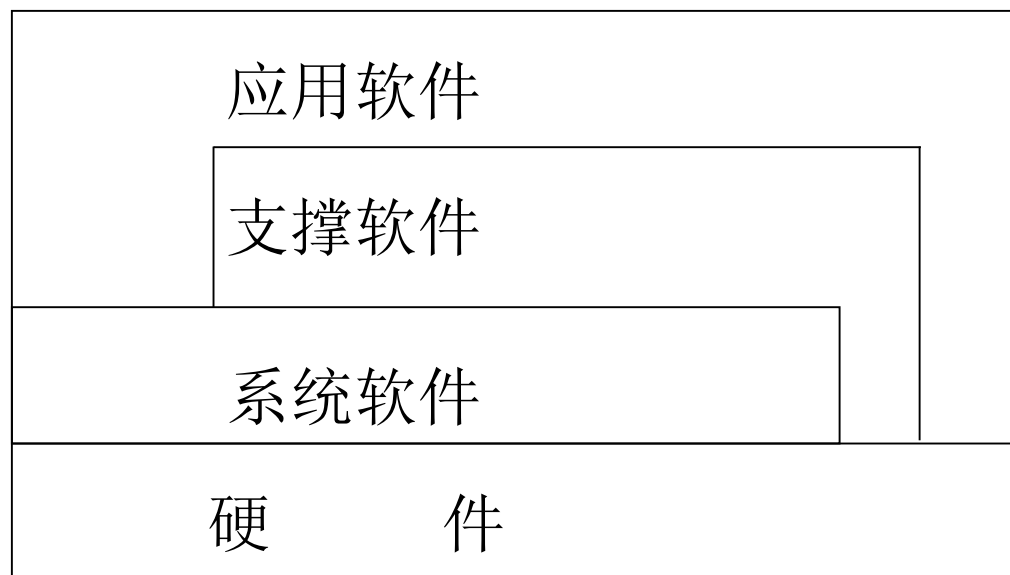
字节: byte (8bit)

代表机型: 4byte (32bit)



❁ 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，有顺序、转移、循环、子程序调用/返回等。

● 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和相应的数据 (data) ，执行一系列操作 (算法, algorithm) ，产生有效的结果。
- 计算 (compute)
 - 数值计算
 - 非数值计算

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

C语言的来历

● **ALGOL 60** (algorithmic language, 国际计算机科学家小组, 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 Martin 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(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语言的实现及其标准

设计



实现

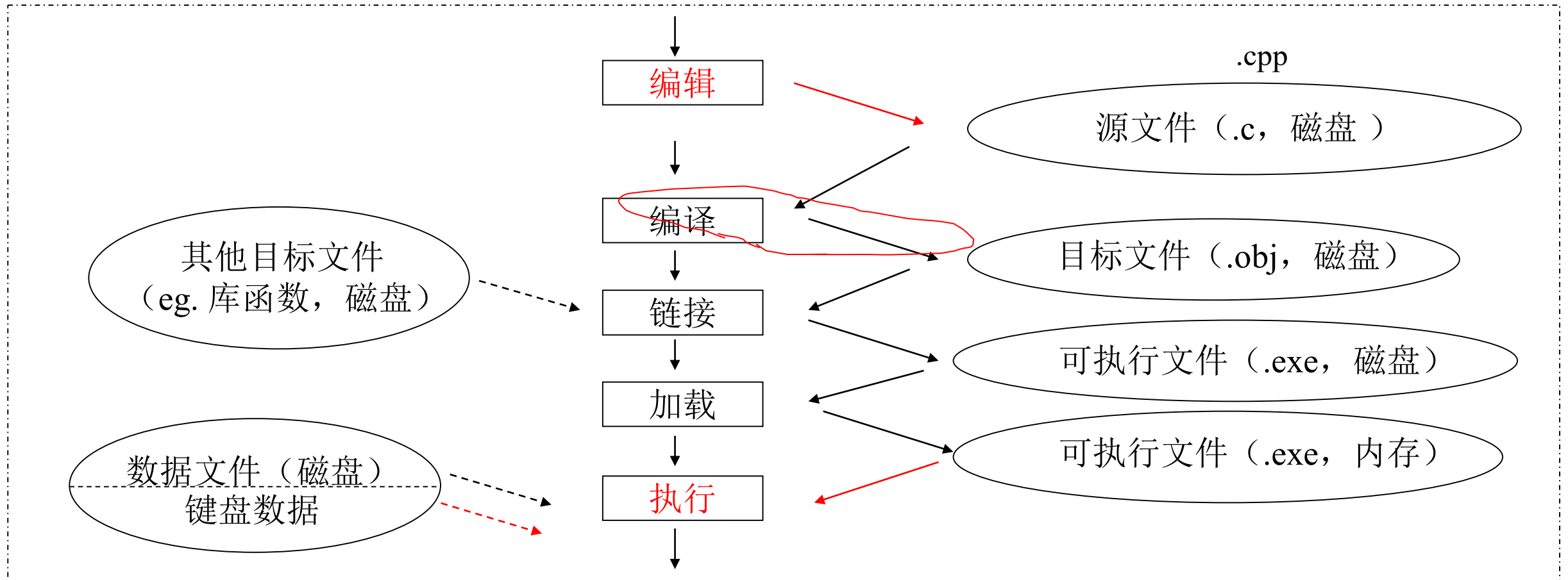


使用

- 1978年 Dennis M. Ritchie 与 Brian W. Kernighan 出版了《The C Programming Language》，此书是最初的C语言“标准”（K&R C）
- 随着C语言的使用和发展，形成了多种C语言的实现版本（implementation），各种版本在功能和函数库的设置内容上存在差别。
- 1983年，ANSI（American National Standards Institute）开始制定统一的C语言标准（specification），直至1989年底正式批准名为ANSI X3.159-1989的标准（C89）
 - 1990年，ISO（International Organization for Standardization）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）
- C17/C18/C2x...
- Stroustrup一直积极推动C++语言的标准，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）
- C++17 ...
- Visual Studio（美国微软公司基于Windows及其相应的硬件平台实现的系列支撑软件）

演示

C程序的开发步骤与集成开发环境 (IDE: Integrated Development Environment)



C程序的基本结构与main函数


一个C程序必须定义一个名字为main的函数

- 红色代码是每一个C程序都应该有的内容（灰色是可以省略的内容）
- 黑色代码是完成特定任务的内容

```
#include <stdio.h>

int main(void)
{
    printf("Now join us!");
    return 0;
}
```

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



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

不好的习惯

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

不好的习惯

standard head

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

C语言程序，可存为.c，也可存为.cpp

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

兼容C语言的C++程序，要存为.cpp

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

C++语言程序，要存为.cpp

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

~ ! # % ^ & * _ - + = | \ : ; “ ‘ , . ? / () { } [] < > Tab(制表) Space(空格) Enter或Return(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

键盘上有的少数符号

汉字库里的符号

比如：

```
#include <stdio.h>

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



没有出现字符集之外的字符

```
#include <stdio.h>

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



出现了字符集之外的字符

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
- 单词与单词之间一般用空格分隔

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。

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

```
#include <stdio.h>

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


标识符 (identifier)

- 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

- 如：include、main、printf...

- 自定义标识符

- 如：变量（程序期间的可变数据）

```
#include <stdio.h>

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

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。

● 字面常量: 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!")

```
#include <stdio.h>

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

C语言的操作符(operator)与标点符号(punctuation)

- 操作符在程序中用来描述对数据的操作，实现运算功能，又叫运算符。

```
#include <stdio.h>

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

- 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
- ...

```
#include <stdio.h>

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

C语言的注释(comment)

🌈 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

```
//This is a C program.  
#include <stdio.h>  
  
int main()  
{    printf("Now Join Us!");  
    return 0;  
}  
/* After the program is executed,  
you will see "Now Join Us!"  
on the display */
```

C语言的语句 (statement)

以分号结尾

```
#include <stdio.h>

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

一个例子

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
    printf("Input n: ");
    scanf("%d", &n);
    while(d <= n)
    {
        sum = sum + 3.14 * d;
        d = d + 1;
    }
    printf("The sum: %f", sum);
    return 0;
}
```

关键字

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
    printf("Input n: ");
    scanf("%d", &n);
    while (d <= n)
    {
        sum = sum + 3.14 * d;
        d = d + 1;
    }
    printf("The sum: %f", sum);
    return 0;
}
```

预定义标识符

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

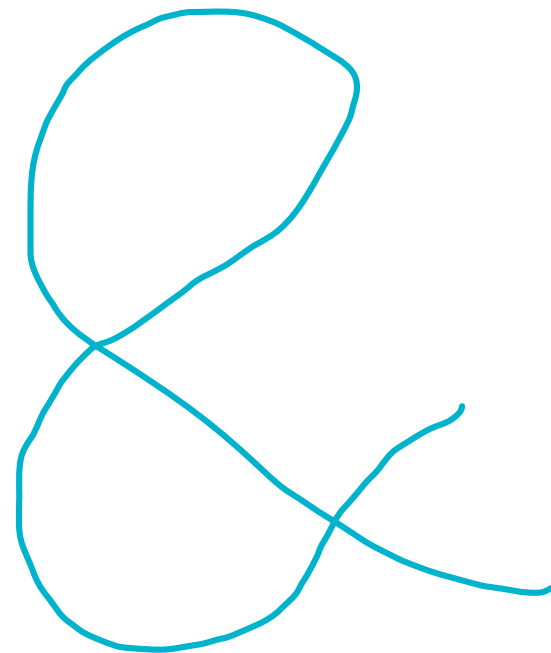
int main()
{
    int n, d = 1;           //d为直径，初始值是1
    double sum = 0;         //sum为圆的周长和，初始值是0
    printf("Input n: ");
    scanf("%d", &n);
    while(d <= n)
    {
        sum = sum + 3.14 * d;
        d = d + 1;
    }
    printf("The sum: %f", sum);
    return 0;
}
```


自定义标识符

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```



字面常量

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```

操作符

//例1.1 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>
```

```
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
```

```
  printf("Input n: ");
```

```
  scanf("%d", &n);
```

```
  while(d <= n)
```

```
  {   sum = sum + 3.14 * d;
```

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

```
  }
```

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

```
  return 0;
```

```
}
```

- 字面常量、变量、...都可以作为操作符的基本操作对象，即操作数 (operand)
- 用操作符将操作数连接起来的式子，叫表达式 (expression)

标点符号

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```

注释

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

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
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```

语句

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

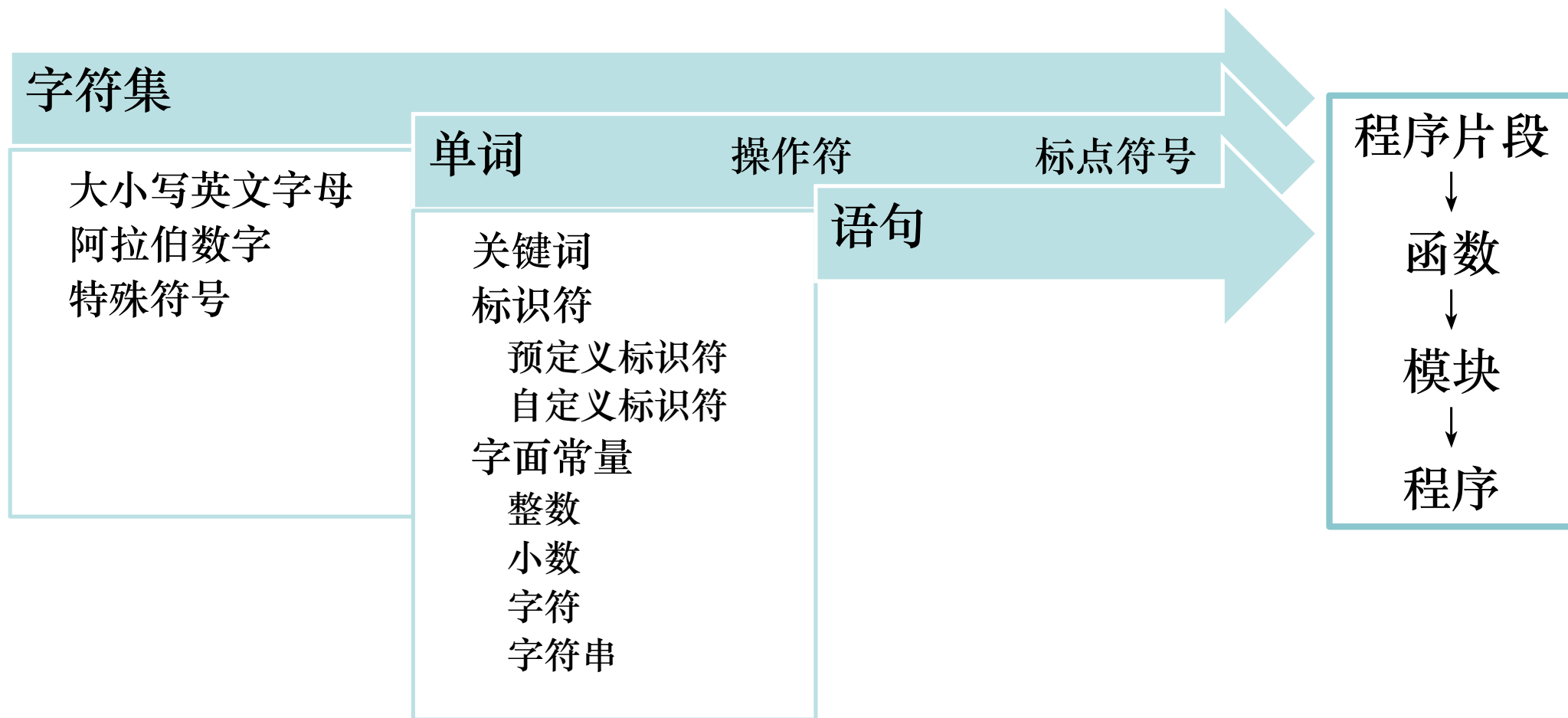
int main()
{ int n, d = 1;          //d为直径，初始值是1
  double sum = 0;        //sum为圆的周长和，初始值是0
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  {   sum = sum + 3.14 * d;
      d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```

```
#include <stdio.h>

int main()
{ int n, d = 1;
  double sum = 0;
  printf("Input n: ");
  scanf("%d", &n);
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{ int n, d = 1;
  double sum = 0;
  cout << "Input n: ";
  cin >> n;
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  cout << "The sum: " << sum;
  return 0;
}
```

初识C语言



变量

- 程序执行期间的可变数据，程序中操作的对象
- 变量的定义（definition）即用数据类型关键字列出变量的类型，并给变量取一个名字
- 变量名是一种典型的自定义标识符
 - 由字符集中的大小写英文字母、阿拉伯数字和下划线组成，且首字符不能是数字
 - 不能与关键字或预定义标识符重复

i、price_car、intStuAge

✓

5x、stu_score、number1、y-average、int、main

✗

变量的定义 (definition)

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

相同类型的多个变量可以并列定义，用逗号分隔

```
{ 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
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    while(d <= n)
```

```
    {    sum = sum + 3.14 * d;
```

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

```
    }
```

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

```
    return 0;
```

```
}
```

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

```
{ int n;
```

```
printf("Input n: ");
```

```
scanf("%d", &n);
```

```
int 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
```

```
while(d <= n)
```

```
{ sum = sum + 3.14 * d;
```

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

```
}
```

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

```
return 0;
```

可以在程序中随时定义变量

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

```
{ int n;
```

```
printf("Input n: ");
```

```
scanf("%d", &n);
```

```
int d = 1;
```

```
double sum = 0;
```

```
while(d <= n)
```

```
{ sum = sum + 3.14 * d;
```

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

```
}
```

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

```
return 0;
```

程序执行含变量定义代码段之前，系统会根据**类型**为变量分配一定大小的空间，以准备存储数据

内存数据区

*****	*****	*****	*****
*****	*****	*****	*****
*****	*****	*****	*****
*****	*****	*****	*****

存储空间里起初是一些0/1组成的无意义的值，可以通过输入或**初始化**来获得有意义的值。

变量的初始化（initialize）

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

```
{ int n;
```

```
printf("Input n:
```

```
scanf("%d", &n);
```

```
int d = 1;
```

```
double sum = 0;
```

```
while(d <= n)
```

```
{ sum = sum + 3.14 * d;
```

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

```
}
```

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

```
return 0;
```

列表初始化:

```
int d = {1};
```

```
int d{1};
```

```
int d(1);
```

内存数据区

000000000	000000000	000000000	00001010
000000000	000000000	000000000	00000001
000000000	000000000	000000000	000000000
000000000	000000000	000000000	000000000

变量的赋值（assignment）

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>

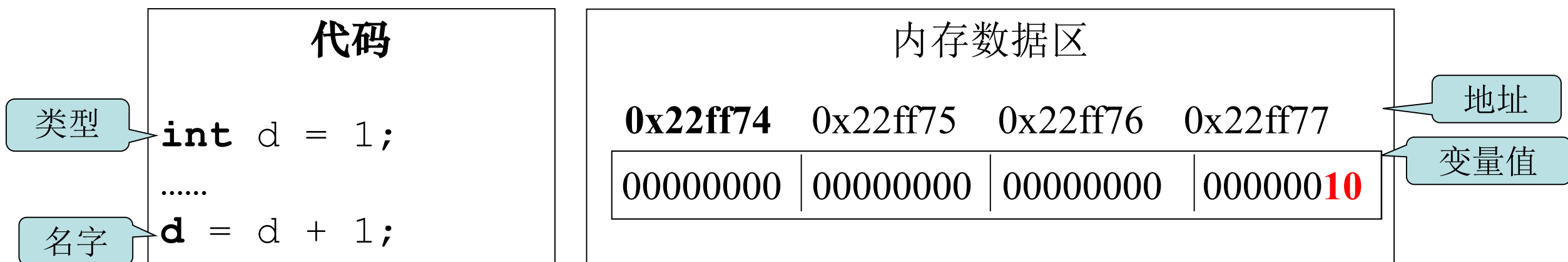
int main()
{ int n;
  printf("Input n: ");
  scanf("%d", &n);
  int d = 1;           //d为直径，初始值是1
  double sum = 0;      //sum为圆的周长和，初始值是0
  while(d <= n)
  { sum = sum + 3.14 * d;
    d = d + 1;
  }
  printf("The sum: %f", sum);
  return 0;
```

变量的值，可以通过赋值来修改。

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←，是将右边的值存入左边变量里。判断两个值是否相等要用两个等于号“==”！

变量的属性

- 变量的内存空间由地址来标识，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。



变量值的输入 (input)

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

```
{ int n;
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    int d = 1;
```

```
    double sum = 0;    //d为直径，初始值是1
```

```
    while(d <= n)
```

```
    {    sum = sum + 3.14 * d;
```

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

```
    }
```

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

```
    return 0;
```

输入格式符

取地址符

scanf_s("%d", &n);

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

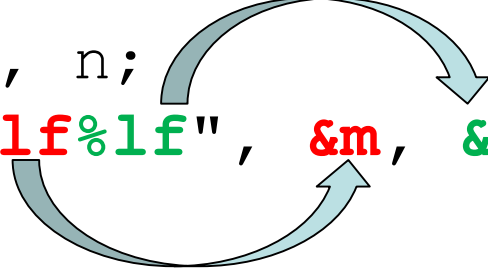
```
    ...
```

```
    cin >> n;
```

```
    ...
```

输入用>>，输出用<<，
莫搞反了！！


```
#include <stdio.h>
int main()
{
    double m, n;
    scanf("%lf%lf", &m, &n);
    .....
    return 0;
}
```



567.1 85.1

567.1
85.1

```
#include <iostream>
using namespace std;
int main()
{
    ...
    cin >> m >> n;
    ...
}
```

数据的输出 (output)

//例1.1 计算一组圆（直径为 n 以内的正整数）的周长之和。

```
#include <stdio.h>
```

```
int main()
```

```
{ int n;
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    int 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
```

```
    while(d <= n)
```

```
    {    sum = sum + 3.14 * d;
```

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

输出格式符

```
    }
```

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

```
    return 0;
```

```
#include <iostream>
```

```
using namespace std;
```

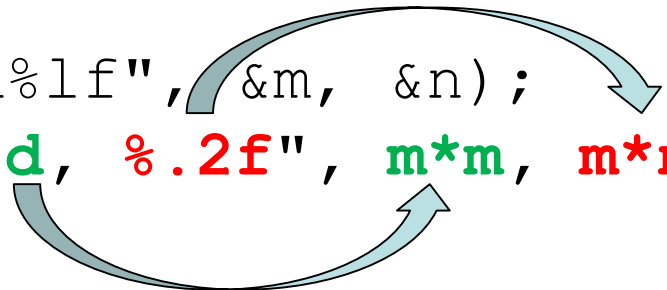
```
int main()
```

```
{ ...
```

```
    cout << "The sum: " << sum;
```

```
    ...
```

```
#include <stdio.h>
int main()
{
    int m;
    double n;
    scanf("%d%lf", &m, &n);
    printf("%d, %.2f", m*m, m*n);
    return 0;
}
```



```
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    ...
    cout << m*m << ", "
    << fixed << setprecision(2)
    << m*n;
    return 0;
}
```

567 85.85
321489, 7370.22

```
#include <stdio.h>
```

如果没有回车换行呢? Try!

```
int main()
```

回车换行转义符 (escape sequence)

```
{
```

```
    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;
```

```
}
```

除法操作符 (两个整数相除,
结果只保留整数部分)

输入

5

输入

8

5 - 8 = -3

5 / 8 = 0

输出

输出

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{ ...
```

注意: 输入 cin 后面不能加 endl

```
    cout << m << " - " << n << " = " << m-n << endl;
```

```
    ...
```

良好的编程习惯

- 设计正确的算法、数据结构与代码
- 采用适合计算机的算法、合理组织数据
- 考虑周全、引入故障检测
- 顾及系统、平台的差异，避免歧义
- 合理抽象、分解、组合
- 提高程序的易读性
 - 注意程序的排版
 - 为程序书写注释
 - 注意自定义标识符的命名风格
 - ...

好的程序：

正确 (correct)
高效 (efficient)
可靠 (reliable)
可移植 (portable)
可重用 (re-usable)
可扩展 (Scalable)
易读 (readability)
.....

排版

- 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）
 - 在操作符两端、逗号后恰当地添加空格
 - 在程序段落之间恰当地添加空行
 - ...

```
#include<stdio.h>
```

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

```
#include<stdio.h>
```

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

一行只写一句！


```
#include<stdio.h>
```

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

缩进！

```
#include <stdio.h>
```

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

操作符两端、逗号后
加空格！

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

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

```
    double sum = 3.14;
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    while(d <= n)
```

```
    {
```

```
        sum = sum + 3.14 * d;
```

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

```
    }
```

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

适当空行！

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

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

```
    double sum = 3.14;
```

```
    printf("Input n: ");
```

```
    scanf("%d", &n);
```

```
    while(d <= n) {
```

```
        sum = sum + 3.14 * d;
```

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

```
    }
```

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

```
    return 0;
```

花括号问题

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
- ...

```
#include <stdio.h>
int main( )
{
    double r, s, l;

    printf("Please input the radius(cm):\n");
    scanf("%lf", &r);
    ...
    printf("The area of the circle: %.2f cm2 ...);
    //Rounded to the nearest hundredth

    return 0;
}
```

```
/**
 *   version4.0:  -----2015-11-10
 *   1、改善画面，数字用颜色区分
 *   2、新增计时和计步功能
 ****/
#include <iostream>
#include <ctime>
...
using namespace std;
void init(int (&board)[4][4]); //初始化棋盘
...
do
{
    a = rand() % 4;
    b = rand() % 4;
} while (arr[a][b] != 0);
arr[a][b] = i; //随机生成2，4
```

自定义标识符命名风格

- 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☆

- 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。
- 【总则】采用一致的、不太长但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。

```
double mj;
```

```
int chicken;
```

- 【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】 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6】 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
```

```
const int MAX_LENGTH = 100;
```

```
const double PI = 3.14;
```

小结

● C程序的组成

字符集
单词
语句
语句块
函数
模块
程序

- **单词**
 - 关键字
 - 标识符
 - 预定义标识符
 - 自定义标识符
 - 字面常量
- 操作符与表达式
- 标点符号

变量（定义、初始化、赋值、值的输入）
变量的属性
数据的输出



要求：

- 会编写简单的C程序
 - 一个程序代码量 \approx 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

Thanks!

