第一章　程序设计基础

刘　卉

huiliu@fudan.edu.cn

主要内容

1.1 程序设计的基本概念

1.2 结构化程序设计

1.3 初识C语言

1.4 高级语言程序开发环境

1.5 C语言的语法基础

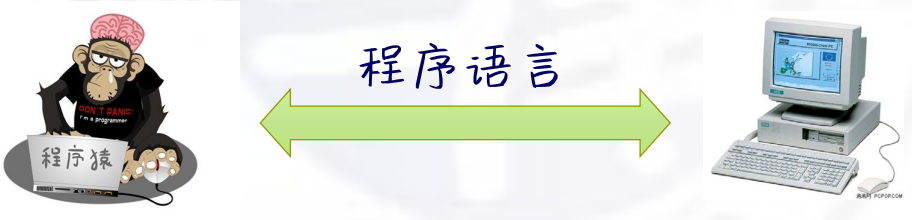
1.6 C程序的书写规范

1.1 程序设计的基本概念

- 程序
- 程序设计
- 程序设计语言
- 算法
- 数据结构

程序, 程序设计, 程序设计语言

- 程序: 用于控制计算机的一系列指令
- 程序设计语言: 描述指令的语言⇨"程序员"与"机器"对话.
  - 语法(syntax): 哪些符号/文字的组合方式是正确的.
  - 语义(semantics): 程序的意义, 程序运行时计算机做什么.



算法 (Algorithm)

问题的求解方法

选择算法的标准

- 正确, 可靠, 简单, 易理解, 时空效率, …

常用工具

- 流程图 (框图)
- 结构化的伪代码 ✓

简单的算法举例

例1.1 求1×2×3×4×5×…×1000

直观解法：

Step1：求1×2，得到结果2.

Step2：将步骤1的结[illegible]

Step3：将步骤2[illegible]到24.

Step4：将步骤3的结果24再乘以5，得到120.

999步？太繁琐！

- 改进的算法
  - 设变量t为因数1，设变量i为因数2，用循环法求结果

Step1 ： 1⇒t

Step2 ： 2⇒i

循环

Step3 ： 使t与i相乘，乘积仍放在变量t中，t×i⇒t

Step4 ： 使i的值加1，即i+1⇒i

Step5 ： 如果i≤n，返回步骤3执行；否则，算法结束

  - 最后得到t的值就是n!的值

求1×3×5×7×9×11

S1：1⇒t

S2：3⇒i

S3：使t与i相乘，乘积仍放在t中，可表示为：t×i⇒t

S4：使i的值加 2，即i+2⇒i

S5：如果i不大于 11 ，返回重新执行S3；否则，算法结束

最后得到t的值就是所求结果

数据结构 (Data Structure)

数据类型

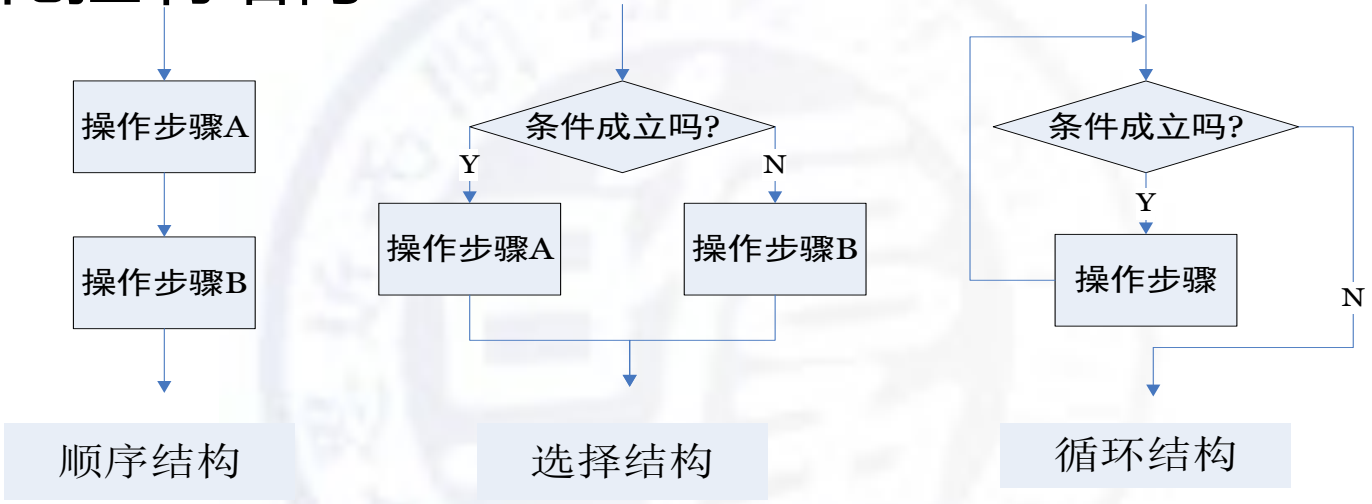
- 变量的取值范围
- 施于变量的操作集合

数据结构

- 数据对象及其相互关系.
- 数据结构与算法紧密相关, 是构造算法的基础.

1.2 结构化程序设计

- 结构化控制结构



  - 三种控制结构可以互相、反复嵌套.
  - 能够描述所有可计算问题.

例1.2 给出一个正整数n(≥3)，判断其是否为素数.

- 所谓素数(prime)，是指除了1和该数本身之外，不能被其它任何整数整除的数.

  例如：13是素数，因为它不能被2，3，4，…，12整除.

- 判断方法：将n作为被除数，将2~n/2 先后作为除数，如果都不能整除，则n为素数.

[算法]

顺序

S1：输入n的值

S2：2⇒i（i作为除数）

S3：n被i除，得余数r

选择

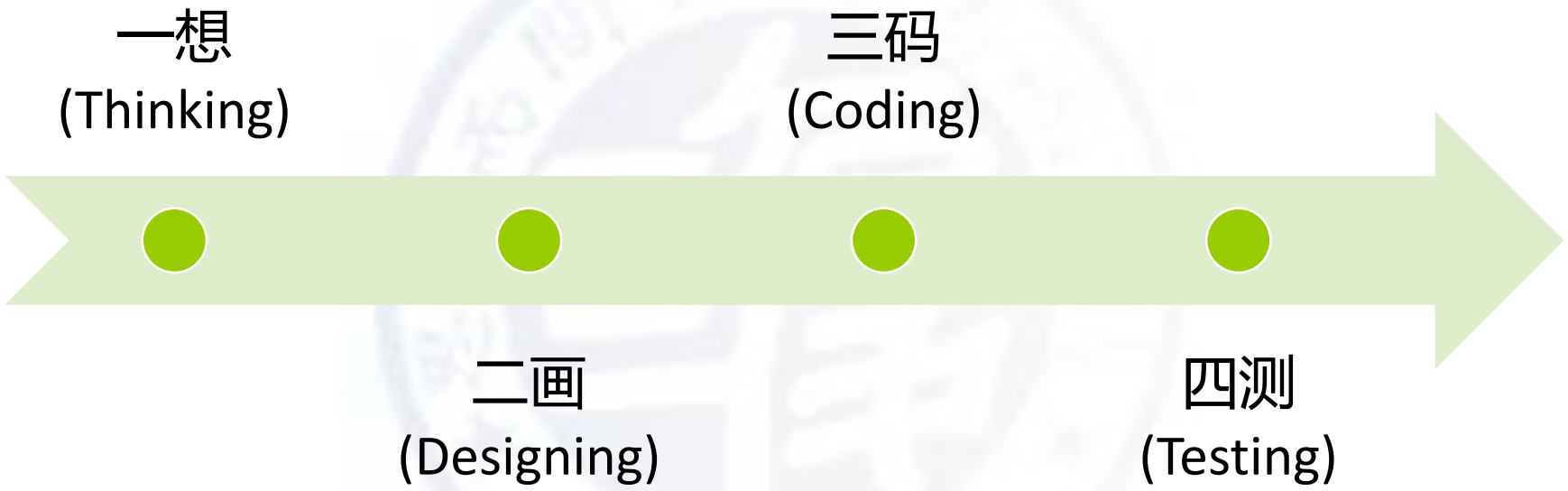
S4：如果r为0，表示n能被i整除，则输出n不是素数，
　算法结束；否则执行S5

S5：i+1⇒i

S6：如果i≤n/2，返回S3；否则输出n是素数，结束.

循环

简单问题的程序设计步骤



- 复杂问题
  - 自顶向下模块化设计方法
  - 模块算法：逐步求精
  - 软件工程

1.3 初识C语言

- 发展过程
  - ALGOL60 ⇨ CPL ⇨ BCPL ⇨ B ⇨ C
  - C标准: C89, C99, C11, C17
- 具有高级语言的特点, 同时可对硬件直接操作

语言特点

C程序的组成

- 一个简单的C程序
  - 编译预处理命令
  - 函数
  - 语句

例1-1 在屏幕上输出一行信息

```c
#include <stdio.h>  /* 编译预处理命令行，stdio.h：输入/输出标准函数
                       库的头文件 */

int main()  // 主函数
{
    printf("To C, or not to C: that is the question.\n");
    // printf(…): 标准输出库函数
    return 0;  // return a value indicating success
}
```

例1-1 在屏幕上输出一行信息

```
#include <stdio.h>  /* 编译预处理命令行，stdio.h：输入/输出
                       标准函数库的头文件 */
int main()  // 主函数
{
    printf("To C, or not to C: that is the question.\n");
    // printf(…): 标准输出库函数
    return 0;  // return a value indicating success
}
```

函数的标志

函数返回值类型

函数体

多行注释

单行注释

C语句的结束符

- 函数是C语言程序的主体.

例1-1 在屏幕上输出一行信息

```
#include <stdio.h>  /* 编译预处理命令行，stdio.h：输入/输出
                       标准函数库的头文件 */
int main()  // 主函数
{
    printf("To C, or not to C: that is the question.\n");
    // printf(…): 标准输出库函数
    return 0;  // return a value indicating success
}
```

函数参数:字符串

换行符

- 双引号是字符串常量的界定符，输出时不显示.
- printf函数不会自动换行

Type in the program carefully

- After you get it to work, please make a few mistakes to see how the tools respond; for example
  - Forget the header
  - Forget to terminate the string
  - Misspell return (e.g., retrun)
  - Forget a semicolon
  - Forget { or }
  - …

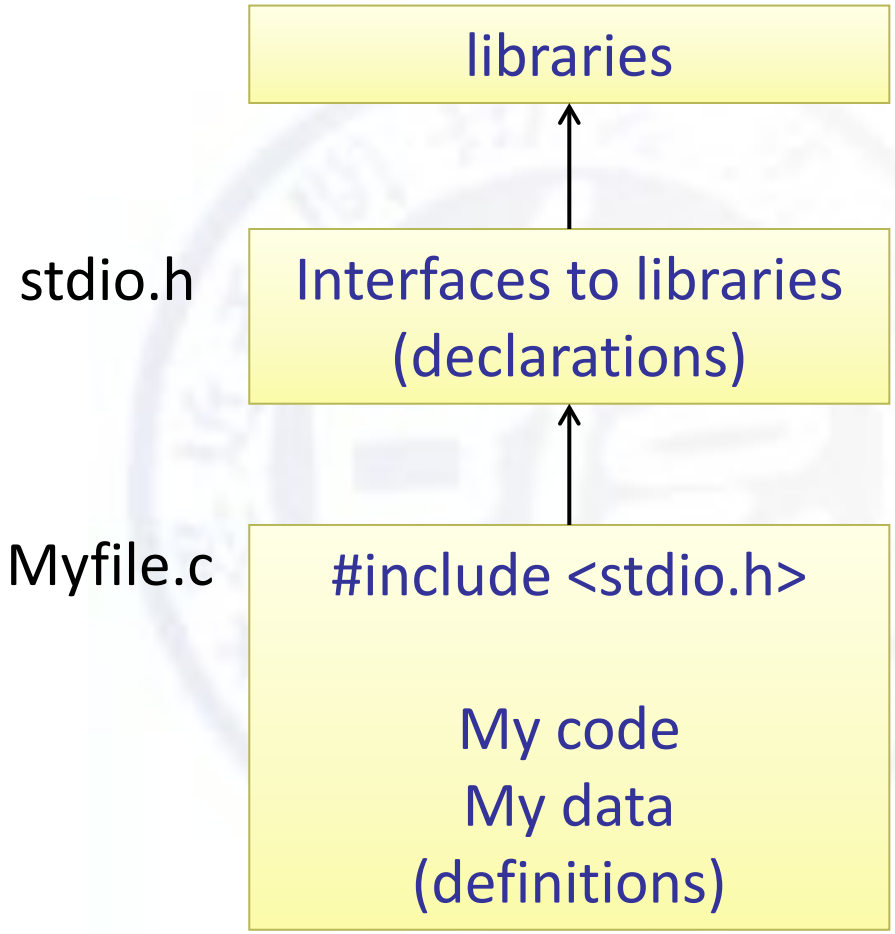
C程序的组成

- 简单程序的一般形式

```
编译预处理命令
int main(void)
{
    语句
    return 0;
}
```

#include <stdio.h>

- 头文件——包含标准库的内容
  - 在编译前, 把stdio.h中的内容包含到程序中.
  - stdio.h包含了C标准输入/输出库的信息. e.g. printf(...)
  - C语言没有内置的读写命令，输入/输出功能由stdio.h中声明的库函数实现.
  - C语言拥有大量类似于stdio.h的头文件(header file).
- 一个编译预处理命令占单独一行, 结尾没有分号


libraries
stdio.h
Interfaces to libraries
(declarations)
Myfile.c
#include <stdio.h>
My code
My data
(definitions)

函数

- C程序就是函数的集合.
- 函数: 完成指定功能的代码
  - 自己编写的函数: 若干语句组合在一起并赋予某个名字.

    e.g. main()函数
  - C标准库提供的函数, 即库函数: 封装了函数代码

    e.g. printf()函数

- C函数 vs 数学函数
  - 相似之处：都能接受不同参数，根据相同的计算方法产生不同的计算结果.
  - 不同之处：C函数不一定有参数，也不一定用来计算数值.
- 函数的调用形式

  函数名(参数1，参数2，…)
- 函数的计算结果：返回值
  - 用return语句返回值.
  - 完成某种功能，不返回值.

- C程序从main()函数开始执行
  - 执行程序时，操作系统会自动调用main函数.
  - 在程序终止时，向操作系统返回一个状态码：main前面的int表明它将返回一个整数值.
  - (void)表明main函数没有参数.
  - "return 0"有两个作用
    - 终止main函数，从而结束程序
    - 指出main函数的返回值是0：表示程序正常终止.

```
int main(void)
{
    语句
    return 0;
}
```

语句

- 程序运行时执行的命令
  - 函数调用语句: 要求某个函数执行分派给它的任务

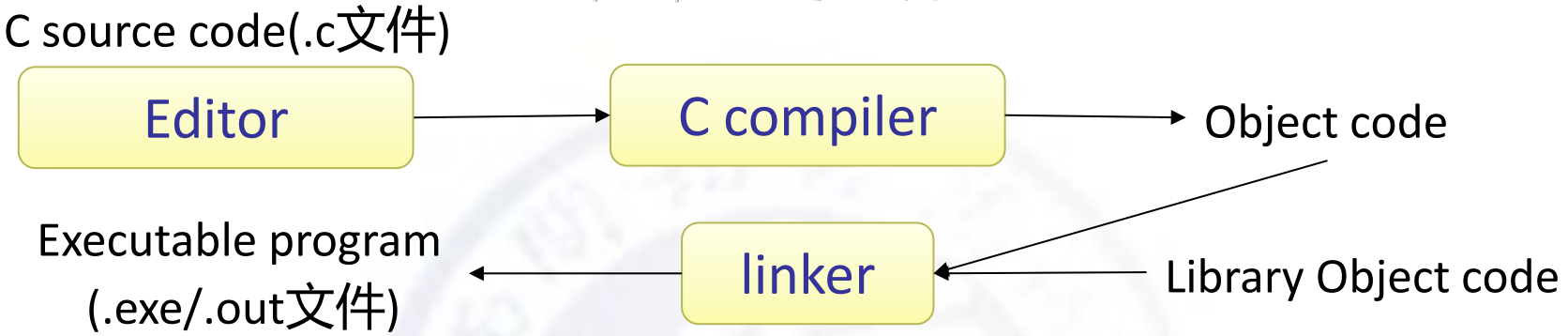
```
e.g.  printf("Hello, world!\n");
```

- 每条语句都必须以分号结束

1.4 高级语言程序开发环境

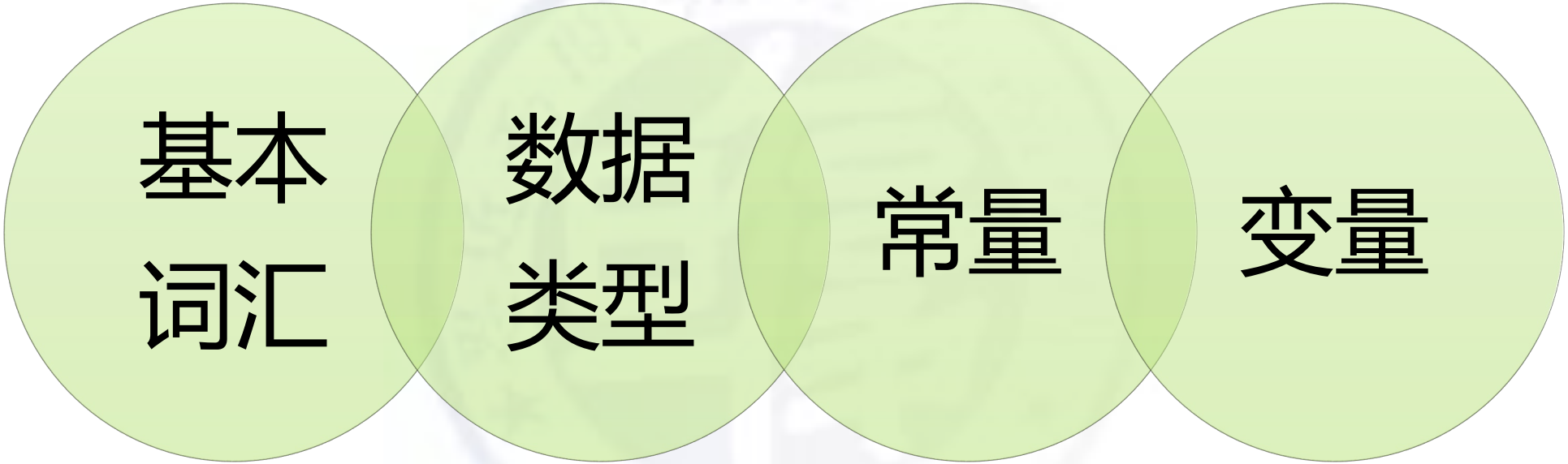
- 集成开发环境(Integrated Development Environment, IDE)
  - IDE是一个软件包：源程序编辑器，编译器，标准函数库，执行程序的运行环境.
  - IDE支持程序从开发到运行的每个阶段：编辑、编译、链接、调试、执行程序.
  - 组成IDE的各个部分协同工作.

    e.g. 当编译器发现程序出错时，它会让编辑器突出显示出错代码行.



- You write C source code
  - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
  - Object code is simple enough for a computer to "understand"
- The linker links your code to system code needed to execute
  e.g., input/output libraries, operating system code, and windowing code
- The result is an executable program
  e.g., an *.exe* file on windows or an *.out* file on Unix

1.5 C语言的语法基础



1.5.1 基本词汇

基本符号

- 数字: 0 ~ 9
- 英文字母: a ~ z, A ~ Z
- 下划线: _
- 特殊符号: %, &, +, -, *, /, …

- 基本词汇：由基本符号组成
  1. 字面形式常量：100，3.14，'a'，"abc"
  2. 特殊符号：+，-，*，/ 等运算符（附录A）
  3. 关键字
     - 英文单词，利用单词的意义表示C程序结构的限定符
     - 程序中不能用关键字命名程序对象
     - 预处理命令：不是关键字，但也不要随意使用

4．标识符

- 用来命名程序对象：变量，常量，类型，函数，语句等.
- 由英文字母或"_"开头，后接0个或多个字母、数字、下划线组成的字符序列.
- 以"_"开头的标识符：系统内部使用.
- 标识符 ⇨ 所标识对象的含义.

  e.g. 定义一个变量保存面积 ⇨ area

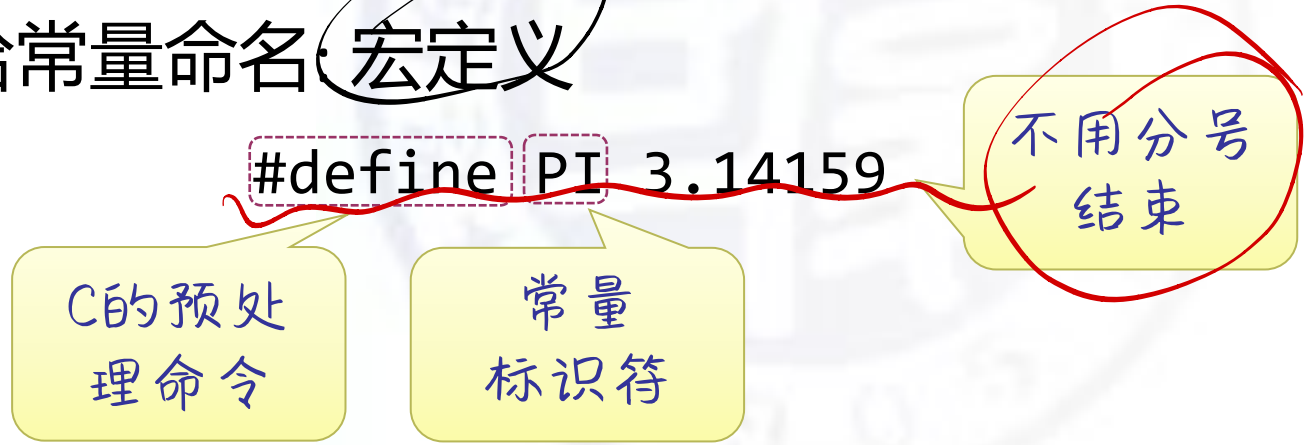
1.5.2 数据类型

- 程序中使用的变量要指定其数据类型
- 基本数据类型
  - 整型, 浮点型, 字符型
- 指针类型
  - 表示数据对象在内存中的地址
- 复杂数据类型
  - 数组, 结构, 联合, 枚举.

1.5.3 常 量

- 程序运行过程中，值不能/不允许改变的数据对象.
- 常量的类型：按值的表示形式区分

  100, 3.14, 'a', "abc"
- 给常量命名：宏定义



1.5.4 变 量

- 程序运行过程中，值可以改变的数据对象.

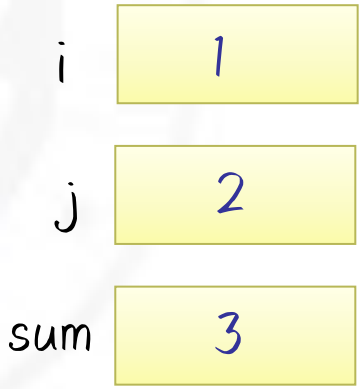
类型　变量名列表;

e.g. `int i, j, sum;`　`/* 定义三个int型变量 */`

编译器分配存储单元

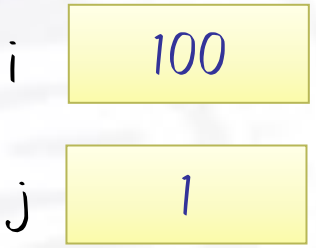
在程序中引用

```
i = 1, j = 2;
sum = i + j;
```



- 变量初始化
  - 定义变量的同时，为变量指定初值

```
int i = 100;
int j = 1;
```



  - 与"先定义后赋值"的区别

```
int i, j;
i = 100;
j = 1;
```

例1-2 计算两个整数的和

```
#include <stdio.h>

int main()
{
    int x, y, sum;  //在程序执行过程中，临时存储数据
                    //变量必须先定义(声明)，后使用.
    printf("Input x and y:\n");
    scanf("%d%d", &x, &y);
    sum = x + y;      //赋值语句
    printf("%d + %d = %d \n", x, y, sum);

    return 0;
}
```


例1-2

为变量输入值

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

- 格式化输入
  - 使用格式字符串指定输入数据的形式
  - %d: 要求读入整数, 变量x, y存储读入的数据
  - % : 引导格式字符
  - & : 后接变量名, 表示取变量的地址.

输出变量/表达式的值

```
printf("%d + %d = %d \n", x, y, sum);
printf("%d + %d = %d \n", x, y, x + y);
```

- 格式化输出
  - %d: 要求输出整数, 变量x, y, sum的值
  - %d还是占位符: 变量/表达式的值的显示位置.
  - \ : 引导转义字符.
  - \n: 1个字符, 终止当前行, 后续输出转到下一行.

例1-3 把F氏温度转换为C氏温度

- 赋值语句
- 循环控制结构
  - 循环控制条件
  - 每执行一次循环, 循环控制量必须改变.
- 使用符号常量而不是"magic numbers"
  - 能提供更多信息, 使程序的阅读和修改都更加容易.


例1-3

C程序的书写规范

- 变量名应反映其含义
- 每个双目运算符的左右加空格，使得运算的结合关系清楚明了
- 每个逗号后加空格
- 缩进有助于轻松识别程序的逻辑结构
- 空行可以把程序划分成逻辑单元，从而使读者更容易辨别程序的结构
- 长语句分行