02 变量与数据类型

程序处理什么数据，又怎样改变它们

本讲学习目标

- 能区分任务数据、变量对象、名称、类型和当前值
- 根据任务数据的含义选择基本类型，说明理由并定义变量
- 解释初始化、有效输入和赋值怎样确定或改变当前值，以及输出怎样读取当前值
- 查用基本输入输出格式，编写或修改简单完整程序
- 用“数据对象 + 操作”说明短程序处理的数据和执行的操作

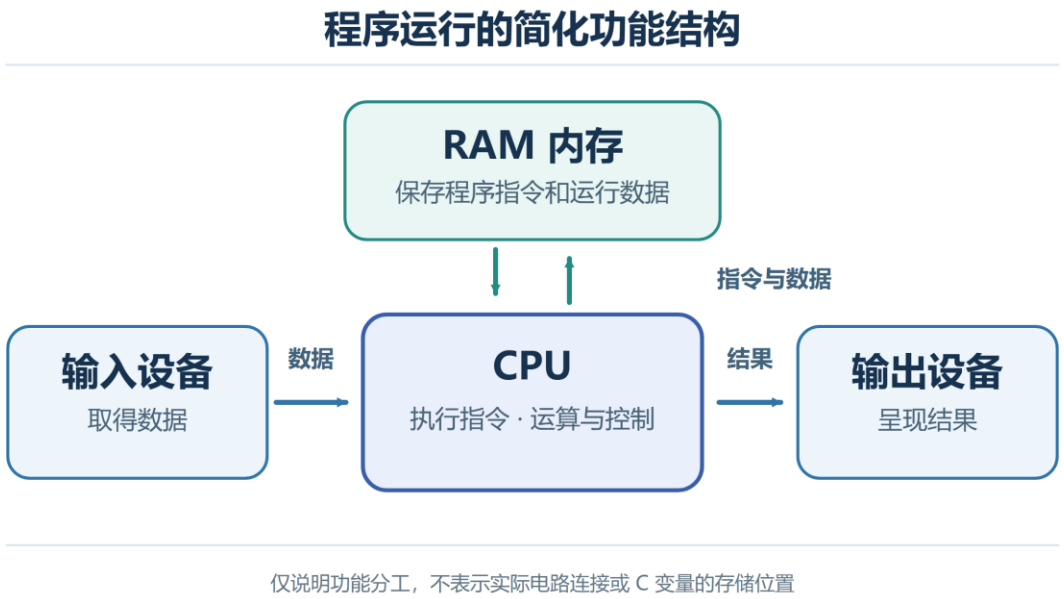
变量：从数据到数据对象

程序 = 数据对象 + 操作

- 变量使程序能够用名称访问具有确定类型的数据对象，保存和使用数据。
- 程序可以读取和改变变量的当前值，处理运行过程中不断变化的数据。

实验记录中的信息处理

- 任务中的信息
  - 样本数、温度和等级
- 计算机中的数据
  - 这些信息用数据表示，供程序处理。
- 处理后的结果
  - 由输出设备呈现



实验记录的数据与操作

- 已有面积程序用变量保存长、宽和面积，操作通过名称访问这些数据。
- 实验记录包含样本数、温度和等级；怎样命名并选择合适类型？
- 从“数据对象 + 操作”观察本讲
  - 对象：用变量保存各项任务数据，通过名称访问。
  - 操作：读取、调整温度、显示结果；下面先区分对象、名称、类型和当前值。
- 变量提供命名访问；它的当前值还取决于已执行的输入或赋值。

1 变量怎样保存数据

先看变量占用的存储空间，再用三种类型的变量保存任务数据

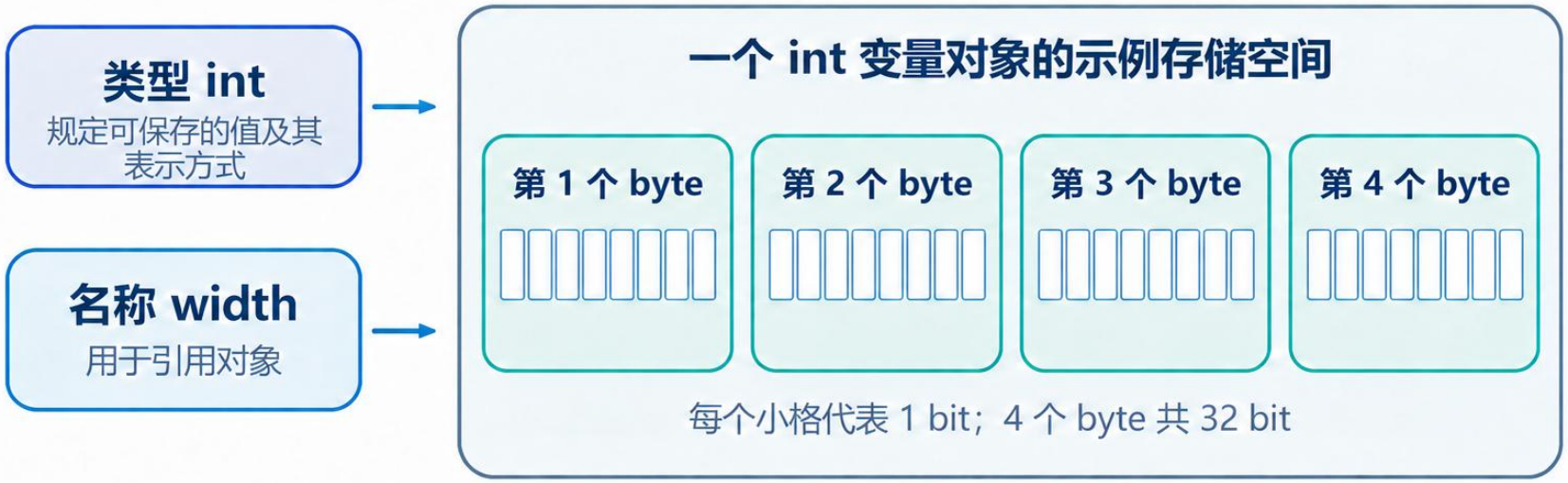
整数变量的存储示意

有类型的变量对象：以 width 为例

示例假设：1 byte = 8 bit，sizeof(int) = 4 byte

bit（二进制位）：每一位可取 0 或 1

byte（字节）：计量存储大小的单位，本图中 1 byte = 8 bit



width 是源码中的名称，不是画在存储空间里的标签

实验记录中的三个变量

```
1 int sample_count = 3;
2 double temperature = 21.5;
3 char quality = 'A';
```

每行定义一个变量对象，并给出初值

整数样本数 → sample_count，初值 3

小数测量温度 → temperature，初值 21.5

单字符质量等级 → quality，初值 'A'

变量的定义与组成

```c
int sample_count = 3;
double temperature = 21.5;
char quality = 'A';
```

变量与定义

变量是程序中具有名称和类型、用于存储数据的对象

变量定义是说明变量的类型与名称、并为对象提供存储的声明

这三行属于声明；C 将声明与语句分开分类

温度变量的组成

类型 double：规定可表示的值及相关操作

名称 temperature：用于在代码中引用该对象

初值 21.5：初始化后，对象的当前值为 21.5

根据数据含义选择类型

整数样本数 → int；小数温度 → double；字符等级 → char

变量定义的基本形式

```
variable-definition ::= type-specifier identifier [ "=" initializer ] ";"
```

- type-specifier：类型说明符；identifier：标识符；initializer：初始化表达式。
  - ::= 表示“定义为”，[ ] 表示可选部分，双引号内表示实际字符。
  - 类型说明符与标识符用空白分隔，定义末尾以分号结束。
- 如：double temperature; 或 double temperature = 21.5;
  - double、temperature、21.5 分别对应类型说明符、标识符、初始化表达式。
- **初始化**是在定义时为对象设置初始值。

标识符与变量命名

```
variable-definition ::= type-specifier identifier [ "=" initializer ] ";"
identifier ::= (letter | "_") { letter | digit | "_" }
```

- **标识符**是用于命名变量、函数等实体的合法字符序列；变量名是一种标识符。
  - letter 为英文字母 A–Z、a–z；digit 为数字 0–9。
  - | 表示“或”，圆括号表示分组，花括号内的部分可重复零次或多次。
  - 标识符区分大小写：count 与 Count 是两个不同的标识符。
  - **关键字**是 C 语言中具有特定用途的保留词，如 int、double，不能用作变量名。
- 命名建议
  - 使用小写字母，以英文字母开头；多个单词用下划线分隔，如 sample_count。
  - 名称体现数据含义和用途。

2 常见类型与值的表示

先区分字面常量与变量，再认识 int、char、double 及值、大小的区别

字面常量与变量对象

```
int sample_count = 3;
double temperature = 21.5;
char quality = 'A';
```

- **字面常量**是在源代码中直接写出、表示一个确定值的写法
  - 3 是整数常量，21.5 是浮点常量，'A' 是字符常量
- 定义建立保存数据的变量对象，名称用于引用它
  - 初始化后，三个对象的当前值分别为 3、21.5、'A'
  - 后续输入或赋值可以改变当前值，不改变源码中的字面常量
- 温度这一行中，类型、名称、字面常量分别在哪里？初始化后的当前值？

数据含义与类型选择

实验记录的信息	数据性质	其他常见例子	本任务使用的类型
样本数	整数数量	人数、次数	int
温度	带小数的测量值	距离、用时	double
等级	单个字符	字母标签	char

- 数据类型决定能保存和处理哪类值
- 选择类型时，要考虑数据含义和可能取值；带小数的数据也可使用 float

常用数据类型

认识整数、字符、浮点数与真假值的表示和基本用法

int 表示一定范围内的整数

```
int sample_count = 3;
```

- int 表示一定范围内的负整数、零和正整数
  - 本任务用它保存整数样本数量；简单输出用 %d
- 内存用二进制位组合表示整数，且是精确的
  - 十进制 3 可写成二进制 11：两位分别表示 2 和 1，合起来为 3
  - 11 是数值的简短写法，不是整个 int 对象的存储长度
  - int 使用有限位数，能表示的整数范围也有限
- int 的具体大小和范围随实现而异

char 用数值编码保存基本字符

```c
char quality = 'A';
```

- **字符编码**规定字符与数值编码之间的对应关系；计算机用这些数值区分字符
  - char 在 C 中属于整数类型
- **ASCII**（美国信息交换标准代码）是一种字符编码，用 0 ~ 127 的编号表示英文字母、数字、标点和控制字符
  - 例如字母 A 的编号是 65，二进制写作 01000001
  - 在使用 ASCII 的环境中，quality 保存这个编码；用 %c 输出时显示为 A
- 一个 char 占一个 C 字节
  - ASCII 有 128 个编码，7 位即可区分；常见的 8 位字节足以保存它们
  - 还有多字节的字符编码，不表示所有文字都只需一个字节
- '\n' 是换行字符常量，表示换行这一控制动作

浮点数为什么可能有误差

```
double temperature = 21.5;
```

- 本任务用 double 保存带小数的测量温度，输出可用 %f
- 常见二进制浮点数用有限位数记录有效数字和指数
  - 指数表示乘以 2 的多少次幂；类似科学记数法中的 $1.25 \times 10^3 = 1250$
- 有些小数可以用有限位数精确表示
  - 21.5 = 16 + 4 + 1 + 0.5，二进制写作 10101.1；小数点后第一位的 1 表示半个单位
- 有些小数需要近似表示
  - 0.1 的二进制小数是 0.000110011…，不能有限写完
  - **舍入**是将所需数值按规则取为可表示的近似值；存储和计算中的舍入可能带来误差

float 与 double：存储、精度与写法

- 常见实现中的存储与精度
  - float：4 字节，约 6 ~ 7 位十进制有效数字；double：8 字节，约 15 ~ 16 位
  - 精度体现能保留多少位有效数字，不是固定的小数位数
  - 例如 12.345 和 0.0012345 都有 5 位有效数字；前导零不计入
- 字面写法
  - 1.2 是 double 常量；后缀 f 使 2.1f 成为 float 常量
  - 1.f 与 1.0f 都表示 float 类型的 1.0，小数点后的数字可以省略
  - 1.2e3 表示 1.2×10^3，即 1200.0；e3 表示乘以 10^3

用整数或 bool 表示真假

```c
#include <stdbool.h>

int ready_as_int = 1;
bool ready = true;
bool finished = false;
```

用整数表示真假

ready_as_int = 1 约定表示“已准备好”；用于条件判断时，0 为假，非零为真

用 bool 明确表示真假

ready = true 表示“已准备好”；finished = false 表示“尚未完成”

C99 引入布尔类型 _Bool；C99～C17 中，<stdbool.h> 提供 bool、true、false 这些名称

类型的大小与范围

用 sizeof 查询大小，再比较重点类型和整数写法的范围

sizeof 查询字节大小

```c
#include <stdio.h>
int main(void)
{
    printf("int: %zu bytes\n", sizeof(int));
    printf("double: %zu bytes\n",
           sizeof(double));
    printf("char: %zu bytes\n", sizeof(char));
    return 0;
}
```

sizeof 给出类型或对象占用的字节数

sizeof(double) 查询类型大小

对前面已定义的 double temperature，sizeof(temperature) 查询对象大小

结果的单位是字节，不读取变量的当前值

%zu：输出大小

sizeof 的结果类型是 size_t，一种用于表示大小的无符号整数类型

printf 用 %zu 将它显示为十进制整数；格式串中的 bytes 是单位标签

当前值与大小

温度从 21.5 改为 22.0 后，对象的类型不变

sizeof(temperature) 会随之改变吗?

常见类型的大小、范围与精度

类型	字节数（常见 8 位字节）	范围 / 精度（常见近似）
char	1（C 标准）	−128 ~ 127 或 0 ~ 255
int	4	约 −2.1e9 ~ 2.1e9
float	4	最大约 3.4e38；6 ~ 7 位有效数字
double	8	最大约 1.7e308；15 ~ 16 位有效数字

除 char 固定为 1 个 C 字节外，表中的字节数和范围都只是常见实现示例。

整数类型的写法与范围

同一类型的写法	常见字节数	范围（常见近似）
short / short int	2	约 −3.3e4 ~ 3.3e4
int / signed int / signed	4	约 −2.1e9 ~ 2.1e9
unsigned int / unsigned	4	0 ~ 约 4.3e9
long / long int	4 或 8	约 ±2.1e9 或 ±9.2e18

- 范围与存储：范围够用时，较小类型可节省大量整数的存储空间；运算未必更快。实际大小与范围依 C 实现而异。
- 符号与用途：差值可能为负，计数通常非负。相同大小的无符号类型有更高的非负上限；计数也可用 int。

3 用变量完成输入、计算与输出

本次任务：输入样本数、温度和单字符等级，将温度增加 0.5 后显示三项结果。

输入输出格式

对象类型	scanf 输入格式	printf 输出格式
int	%d	%d
float	%f	%f
double（本任务温度）	%lf	%f
char	%c	%c

- 输出温度：%.1f 显示一位小数；%.2f 显示两位。
- 输入：用 &变量名 指定读入哪个普通变量；格式串中 %c 前的空格跳过字符前的空格、换行等空白字符。

实验记录程序

```c
#include <stdio.h>
int main(void)
{
    int sample_count = 0;
    double temperature = 0.0;
    char quality = '?';
    scanf("%d%lf %c",
          &sample_count,
          &temperature, &quality);
    temperature = temperature + 0.5;
    printf("count=%d, "
           "temperature=%.1f, "
           "quality=%c\n",
           sample_count, temperature,
           quality);
    return 0;
}
```

保存三项数据 → 定义变量

int 保存样本数，double 保存温度，char 保存等级

取得本次数据 → 输入

格式顺序与三个目标变量对应

修正温度 → 赋值

temperature 增加 0.5，样本数和等级不变

显示三项结果 → 输出

输出格式与三个当前值对应

输入、更新与输出

执行位置	sample_count	temperature	quality
输入后	3	21.5	'A'
温度更新后	3	22.0	'A'

- 有效输入为 3 21.5 A；赋值只改变温度，输出读取当前值。
- 预期输出：count=3, temperature=22.0, quality=A

练习：输入并显示运动记录

```c
#include <stdio.h>

int main(void)
{
    /* 在此定义任务所需变量 */
    /* 在此读取本次数据 */
    /* 在此显示记录 */

    return 0;
}
```

任务

输入整数圈数、带小数的总用时和单字符等级

原样显示本次输入的三项数据

有效输入示例：4 62.5 B

任务分析

确定需保存的三项数据

指出实验记录中的哪一步处理不再需要

程序补全

自选类型与名称，补全定义、输入和输出；写出输入 4 62.5 B 后的预期显示

解析：输入并显示运动记录

```c
#include <stdio.h>
int main(void)
{
    int lap_count = 0;
    double time_seconds = 0.0;
    char grade = '?';
    scanf("%d%lf %c", &lap_count,
          &time_seconds, &grade);
    printf("laps=%d, time=%.1f, grade=%c\n",
           lap_count, time_seconds,
           grade);
    return 0;
}
```

参考实现

int 表示圈数，double 表示总用时，char 表示等级

scanf 的 %lf 对应 double 目标；printf 的 %.1f 显示一位小数

预期结果

输入 4 62.5 B，显示 laps=4, time=62.5, grade=B

本讲总结：变量对象与操作

本讲用“数据对象 + 操作”理解实验记录和运动记录。

- 数据对象：变量按任务命名，提供保存和访问数据的对象；类型限定可表示的值。
- 值与大小：字面常量可用于初始化；sizeof 给出字节大小，不是当前值。
- 操作：输入写入、赋值更新、输出读取；通过变量名把处理与对应对象联系起来，格式与类型对应。
- 结果：当前值取决于执行位置；预期结果与实际运行结果要区分。

教材阅读指引

教材位置	本讲使用方式
《C Primer Plus》第六版第 2 章	回看声明、变量与赋值在完整短程序中的作用
第 3 章 3.2～3.6 节	重点阅读整数 / 浮点区别、int、char、double、变量与常量及类型大小的相关内容
第 4 章 4.4 节	查阅本讲使用的 scanf()、printf() 格式说明符
背景与暂不要求	第 1 章的计算机结构只作背景；字符串、完整类型目录、输入失败处理及系统调试不属于本讲学习要求