02 变量与数据类型

程序怎样保存和改变数据

本单元的学习目标

- 说明变量、变量名、数据类型和当前值之间的关系
- 根据数据含义选择 int、double 或 char
- 追踪赋值和输入前后的变量状态
- 发现并修正变量使用和输入输出中的常见错误

1 从状态变化到第一个变量

先观察值怎样随执行变化，再建立并追踪第一个变量

矩形程序的三个时刻

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    scanf("%d%d", &width, &height);
    area = width * height;
    printf("Area = %d\n", area);

    return 0;
}
```

执行时刻	width	height	area
初始化完成	0	0	0
输入 4 6 后	?	?	?
面积赋值后	?	?	?

先填表，再运行核对

当前值与程序状态

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    scanf("%d%d", &width, &height);
    area = width * height;
    printf("Area = %d\n", area);

    return 0;
}
```

执行时刻	width	height	area
初始化完成	0	0	0
输入 4 6 后	**4**	**6**	0
面积赋值后	4	6	**24**

执行位置不同，变量当前值可能不同

需要保存的样本数量

- 程序需要记录已经取得多少个样本
- 这个数据是整数计数
- 程序开始时，计数是 0

第一个变量

```c
#include <stdio.h>

int main(void)
{
    int sample_count = 0;

    printf("before: %d\n", sample_count);
    sample_count = 3;
    printf("after: %d\n", sample_count);

    return 0;
}
```

数据需要

保存整数样本数量

`int`

选择表示整数的类型

`sample_count`

名称说明变量用途

`0`

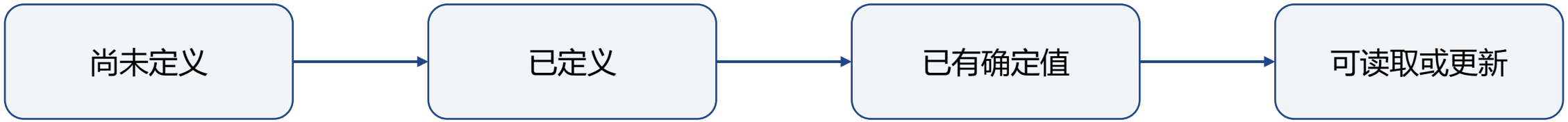
建立第一个确定值

变量

- **定义**：变量是程序运行期间保存一个值，并允许该值发生改变的对象。
- **示例**：int sample_count = 0;
  - int：变量的数据类型
  - sample_count：源代码引用变量的名称
  - 0：变量建立时的初值
- **当前边界**：变量、变量名和当前值相互关联，但不是同一个概念。

名称可用与值可读取

- 定义使名称从相应位置开始可用
- 初始化、成功输入或赋值可以建立确定值
- 读取变量以前，两个条件都要满足



赋值前后的预测

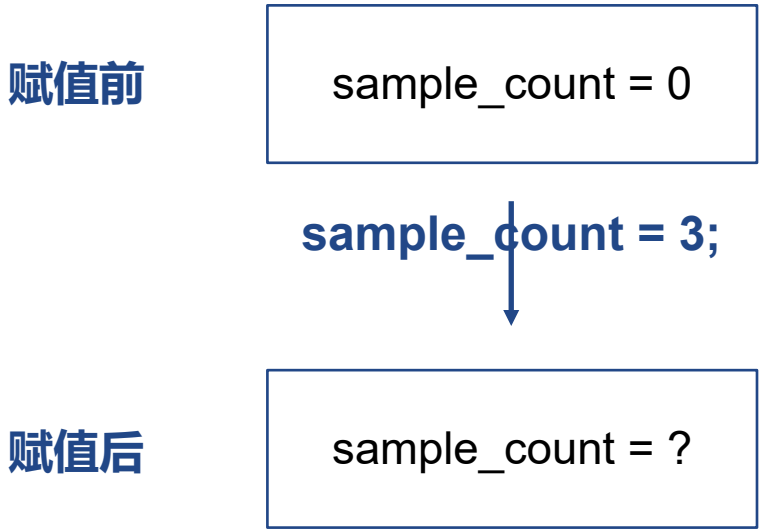
sample_count.c

```c
#include <stdio.h>

int main(void)
{
    int sample_count = 0;

    printf("before: %d\n", sample_count);
    sample_count = 3;
    printf("after: %d\n", sample_count);

    return 0;
}
```



先预测，再运行

赋值改变当前状态

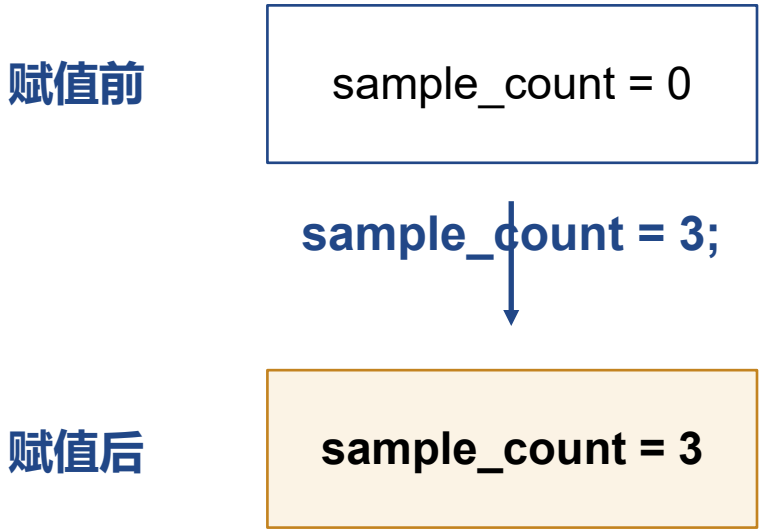
sample_count.c

```c
#include <stdio.h>

int main(void)
{
    int sample_count = 0;

    printf("before: %d\n", sample_count);
    sample_count = 3;
    printf("after: %d\n", sample_count);

    return 0;
}
```



旧状态 + 当前语句 = 新状态

单变量检查

- **建立**：补全 int sample_count = 0;
- **预测**：判断 sample_count = 3; 执行后的值
- **解释**：区分变量名、数据类型和当前值

2 从一个变量到多种数据

数据含义不同，变量需要选择合适的类型和使用方式

三种数据需求

数据含义	示例值	程序需要怎样解释
样本数量	3	整数计数
测量温度	21.5	带小数的测量值
质量等级	'A'	单个字符

先辨认数据含义，再决定怎样表示

数据类型

- **数据类型**规定对象可以表示哪类值，以及相关操作怎样解释这些值。
- 类型是变量对象的属性，不是变量的当前值
- 选择类型要依据数据含义和后续操作
- 本单元只使用 int、double 和 char 的常见入门职责

依据数据含义选择类型

数据含义	示例值	当前选择
整数计数	3	int
带小数的测量值	21.5	double
单个字符	'A'	char

三个变量

```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;
}
```

sample_count
整数计数，初值 0

temperature
带小数的测量值，初值 0.0

quality
单个字符等级，初值 '?'

变量何时可以读取

代码动作	名称是否可用	是否已有确定值
尚未定义	否	否
int count;	是	否
int count = 0;	是	是
成功输入或赋值以后	是	是，值可能已经改变

两类使用顺序错误

定义前使用

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("%d\n", sample_count);
6     int sample_count = 0;
7
8     return 0;
9 }
```

名称尚不可用

定义后直接读取

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int sample_count;
6     printf("%d\n", sample_count);
7
8     return 0;
9 }
```

尚未建立确定值

类型与格式说明符

查用项	最小写法	当前用途	边界或提醒
int	0	输入、输出均用 %d	整数计数
double	0.0	输入用 %lf，输出用 %f	带小数的测量值
char	'?'	输入、输出均用 %c	输入时，%c 前的空格可跳过此前留下的空白
scanf 的 &	&变量名	帮助输入定位变量	当前只按模板使用

输入必须指向相应变量

- scanf 根据格式说明符解释输入数据
- &变量名 指明接收输入的变量
- 本单元先把 &变量名 作为输入模板使用，指针含义留到后续单元

3 综合追踪、修错与迁移

用已有模型追踪完整程序，并独立完成修错和新任务

实验记录程序

experiment_record.c

```
1 #include <stdio.h>
2 int main(void)
3 {
4     int sample_count = 0;
5     double temperature = 0.0;
6     char quality = '?';
7     scanf("%d%lf %c",
8           &sample_count, &temperature,
9           &quality);
10    temperature = temperature + 0.5;
11    printf("count=%d, temperature=%.1f, quality=%c\n",
12           sample_count, temperature, quality);
13    return 0;
14 }
```

输入后的状态预测

experiment_record.c

```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;
}
```

执行时刻	sample_count	temperature	quality
输入后	?	?	?

课堂输入：3 21.5 A

先预测，再运行

输入建立新的当前值

experiment_record.c

```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;
}
```

执行时刻	sample_count	temperature	quality
输入后	**3**	**21.5**	**'A'**

成功输入为三个变量建立新的确定值

赋值后的状态

experiment_record.c

```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;
}
```

执行时刻	sample_count	temperature	quality
输入后	3	21.5	'A'
校正后	3	**22.0**	'A'

当前语句只更新 temperature

怎样追踪变量状态

- **定位**：确定即将执行的语句及其涉及的变量
- **推演**：写出该语句执行后的当前值
- **核对**：用输出或调试观察检验预测

变量建立错误

错误 A

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("%d\n", sample_count);
6     int sample_count = 0;
7
8     return 0;
9 }
```

错误 B

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int sample_count;
6     printf("%d\n", sample_count);
7
8     return 0;
9 }
```

1 判断最早断点　2 修改代码　3 说明依据

类型与格式不匹配

```c
#include <stdio.h>

int main(void)
{
    double temperature = 0.0;
    scanf("%f", &temperature);
    printf("%f\n", temperature);

    return 0;
}
```

变量类型

temperature 是 double

当前输入格式

%f 与 double 不匹配

查表修正

double 输入使用 %lf

再次验证

重新构建并运行

运动测试记录

- **需要保存的数据**
  - 完成的圈数
  - 用时，单位为秒
  - 本次测试的单字符等级
- **编程任务**
  - 根据数据含义确定变量的类型、名称和初值
  - 完成一次输入或赋值，并输出结果

语言模型与执行实现

- 在 C 语言层，用变量、数据类型和当前值解释程序行为
- 构建后的机器指令由 CPU 执行
- RAM 为运行中的程序和数据提供工作空间
- 本单元的状态图用于追踪 C 程序，不表示真实的 RAM 布局

本单元总结

核心认识	判断标准
变量	变量是保存一个值并允许该值改变的对象；变量名用于引用它
数据类型	数据类型规定对象可以表示的值及相关操作的解释方式
程序状态	当前执行位置与相关变量的当前值共同支持状态追踪
状态变化	定义并初始化、赋值和输入都可能建立或改变变量的当前值
核查方法	先预测，再执行，并用输出、状态表或调试观察核对