第4讲　顺序结构程序设计

目录

4.1 标准输入输出函数

标准输入输出函数

- **C标准库提供了基本输入输出功能**
- **要使用这些功能，需要包含<stdio.h>头文件**

常用的输入/输出函数包括：

`getchar()`	`putchar()`	单个字符输入/输出函数
`scanf()`	`printf()`	格式化输入/输出函数
`gets()`	`puts()`	字符串输入/输出函数

单个字符输入/输出函数

getchar()

- 作用：从键盘读取**一个字符**，返回其**ASCII码值**（返回值为int型）
- 当连续输入了多个字符时，只能读取第一个字符
- 回车也算输入字符
- 多次调用该函数，能够实现读取多个字符

putchar()

- 作用：在控制台输出**一个字符**
- 需接收1个参数，该参数是需要输出的字符的ASCII码值（char类型、int类型均可）
- 多次调用该函数，能够输出多个字符

```c
#include <stdio.h>

int main()
{
    printf("Please input a character: ");
    char c = getchar();
    printf("The character you input is: ");
    putchar(c);
    putchar('\n');
    return 0;
}
```

单个字符输入/输出函数

```c
#include <stdio.h>

int main()
{
    char c1, c2;

    printf("本软件带有x插件，安装否(y/n)? ");
    c1 = getchar();
    if (c1 == 'Y' || c1 == 'y')
        printf("正在安装插件x...安装完毕! ");

    printf("\n本软件带有y插件，安装否(y/n)? ");
    c2 = getchar();
    if (c2 == 'Y' || c2 == 'y')
        printf("正在安装插件y...安装完毕! ");

    return 0;
}
```

getchar的错误用法

假设某软件包含x和y两个插件，需要分别用户分两次输入y/n，分别决定是否安装每个插件。左边的代码片段能否实现这一功能需求?

分析:

- 在用户决定是否安装x插件时，用户输入y（或者n）并按下回车（\n）；
- 此时用户的键盘输入被发送到缓冲区；
- 第1次调用的getchar()，获取了用户输入的字母y，之后字母y从缓冲区中删去，缓冲区还剩下字符\n；
- 第2次调用getchar()，获取了缓冲区剩下的字符\n，用户无法再次输入以决定是否安装y插件。

单个字符输入/输出函数

```c
#include <stdio.h>

int main()
{
    char c1, c2;

    printf("本软件带有x插件，安装否(y/n)? ");
    c1 = getchar();
    if (c1 == 'Y' || c1 == 'y')
        printf("正在安装插件x...安装完毕! ");

    getchar();

    printf("\n本软件带有y插件，安装否(y/n)? ");
    c2 = getchar();
    if (c2 == 'Y' || c2 == 'y')
        printf("正在安装插件y...安装完毕! ");

    return 0;
}
```

改进：中间再调用一次getchar()，以清除缓冲区！

单个字符输入/输出函数

“复读机”：

```c
#include <stdio.h>

int main()
{
    while(1) {
        putchar(getchar());
    }
    return 0;
}
```

“竖排文字”：

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

int main(int argc, char const *argv[])
{
    char str[] = "Hello World";
    for (int i = 0; i <= strlen(str); i++) {
        putchar(str[i]);
        putchar('\n');
    }
    return 0;
}
```

格式化输入/输出函数

scanf()

- 作用：按照一定的格式读取键盘输入
- 用法：scanf(格式控制字符串，地址列表)
- 一次能够读取一组键盘输入，每个输入的格式是由格式控制字符串决定的

printf()

- 作用：按照一定的格式在控制台输出数据
- 用法：printf(格式控制字符串，输出列表)
- 一次能够按规定格式输出一组数据，每个输出数据的格式是由格式控制字符串决定的

```c
#include <stdio.h>

int main()
{
    char name[50];
    int age;
    float height, weight;

    printf("Please enter your name, age, height(cm)"
           " and weight(kg): ");
    scanf("%s%d%f%f", name, &age, &height, &weight);
    printf("Your name: %s\n"
           "Your age: %d\n"
           "Your height: %.2fcm\n"
           "Your weight: %.2fkg",
           name, age, height, weight);

    return 0;
}
```

格式化输入/输出函数

格式控制字符

形式：%[flags][width][.precision][length]specifier

specifier（格式字符）	意义
d	以十进制形式输出带符号整数
o	以八进制形式输出无符号整数
x或X	以十六进制形式输出无符号整数
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e或E	以指数形式输出单、双精度实数
c	输出单个字符
s	输出字符串
p	输出指针地址

格式化输入/输出函数

格式控制字符

形式：%[flags][width][.precision][length]specifier

flags（标识）	描述
-	在给定的字段宽度内左对齐，默认是右对齐（参见 width 子说明符）。
+	强制在结果之前显示加号或减号（+ 或 -），即正数前面会显示 + 号。默认情况下，只有负数前面会显示一个 - 号。
空格	如果没有写入任何符号，则在该值前面插入一个空格。
#	与 o、x 或 X 说明符一起使用时，非零值前面会分别显示 0、0x 或 0X。
0	在指定填充 padding 的数字左边放置零（0），而不是空格（参见 width 子说明符）。

格式化输入/输出函数

格式控制字符

形式：%[flags][width][.precision][length]specifier

width（宽度）	描述
(number)	要输出的字符的最小数目。如果输出的值短于该数，结果会用空格填充。如果输出的值长于该数，结果不会被截断。
*	宽度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。

.precision（精度）	描述
.number	对于整数说明符（d、i、o、u、x、X）：precision 指定了要写入的数字的最小位数。如果写入的值短于该数，结果会用前导零来填充。如果写入的值长于该数，结果不会被截断。精度为 0 意味着不写入任何字符。
	对于 e、E 和 f 说明符：要在小数点后输出的小数位数。
	对于 s：要输出的最大字符数。默认情况下，所有字符都会被输出，直到遇到末尾的空字符。
	对于 c 类型：没有任何影响。
	当未指定任何精度时，默认为 1。
.*	精度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。

格式化输入/输出函数

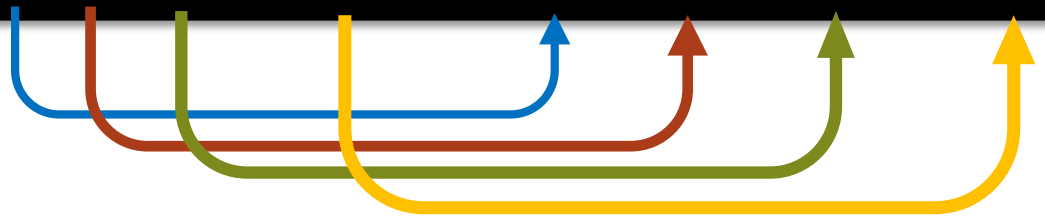
格式控制字符

形式：%[flags][width][.precision][length]specifier

length（长度）	描述
h	参数被解释为短整型或无符号短整型（仅适用于整数说明符：d、o、u、x 和 X）。
l	参数被解释为长整型或无符号长整型，适用于整数说明符（d、o、u、x 和 X）及说明符 c（表示一个宽字符）和 s（表示宽字符字符串）。
L	参数被解释为长双精度型（仅适用于浮点数说明符：e、E、f）。

格式化输出函数：printf

```
printf("%s %d %.2fcm %.2fkg", name, age, height, weight);
```



注意事项：

- 格式控制字符串跟普通字符串一样需要用**双引号**括起来
- 使用 printf 时，格式控制字符的数量与输出列表中的元素数量必须一致，且严格一一对应
- 输出时，格式控制字符的位置将会被输出列表中对应的表达式的值替换
- 对于字符串中的非格式控制字符（例如上面的例子中的cm、kg），原样输出
- 如果字符串没有任何格式控制字符，那么可以没有输出列表，printf相当于一个普通字符串输出函数。

格式化输出函数：printf

printf 最常见的用法：

- **输出规整的表格数据（每列等宽，配置合适的对齐方式）**
- **输出浮点数时，限定输出的小数位数**

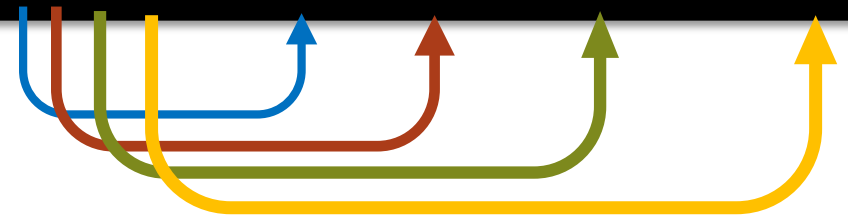
```c
#include <stdio.h>

int main(int argc, char const *argv[])
{
    printf("%-20s%15s%15s\n", "国家或地区", "面积（平方公里）", "人口（千人）
");
    printf("%-20s%15d%15.2f\n", "中华人民共和国", 9600000, 1370536.);
    printf("%-20s%15d%15.2f\n", "蒙古国", 1566500, 2353.);
    printf("%-20s%15d%15.2f\n", "马来西亚", 329758, 21169.);
    printf("%-20s%15d%15.2f\n", "新加坡共和国", 641, 3100.);
    return 0;
}
```

输出结果

```
国家或地区              面积（平方公里）      人口（千人）
中华人民共和国              9600000    1370536.00
蒙古国                      1566500       2353.00
马来西亚                     329758      21169.00
新加坡共和国                    641       3100.00
```

格式化输入函数：scanf

```
scanf("%s%d%f%f", name, &age, &height, &weight);
```



注意事项：

- 格式控制字符串跟普通字符串一样需要用**双引号**括起来
- 使用 scanf 时，格式控制字符的数量与地址列表中的元素数量必须一致，且严格一一对应
- 对于字符串中的非格式控制字符（例如下面的例子中的name=、逗号等），需要**原样输入，一字不差（如果是空格，那么换行符、制表符等其他空白字符也等价于空格）**

```
scanf("name=%s,age=%d,height=%f,weight=%f", name, &age, &height, &weight);
```

- 地址列表中必须是**变量的地址**，不要漏掉取址符&；
- **【特殊】**对于**数组**来说，由于数组变量本身就是指向数组首元素的地址，所以不能画蛇添足给数组前面加&；
- scanf 函数的返回值是**成功读取数据的个数**，如果没有读取任何数据，则返回**-1 (EOF)**

格式化输入函数：scanf

```
scanf("%s%d%f%f", name, &age, &height, &weight);
```

关于输入数据的分隔：

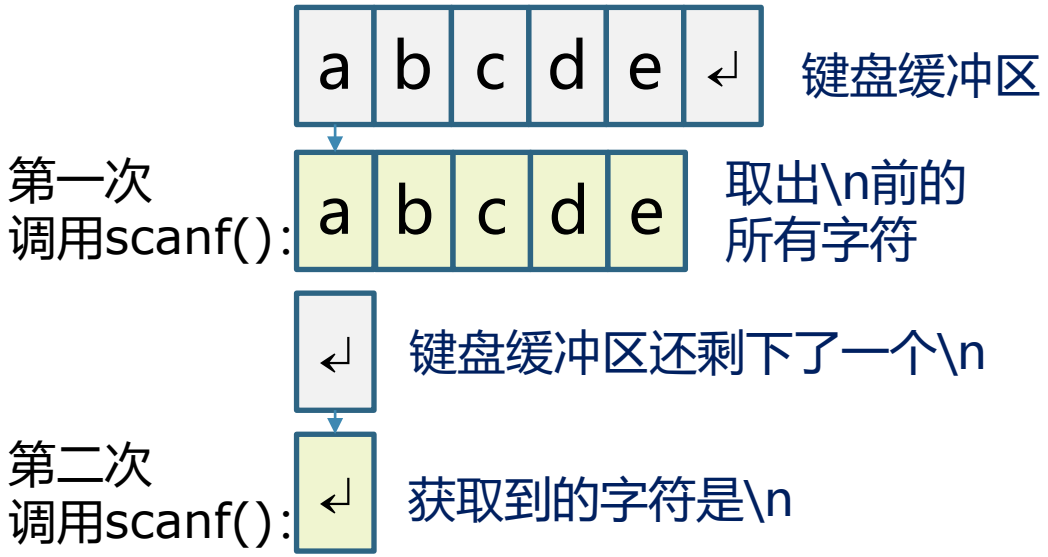
- 通常，数据之间需用一个或以上的空格、换行（\n）或制表符（\t）等空白字符隔开，例如输入一个数字，这个数字前面输入的所有空格、换行都会被忽略；
- 如果数据是字符串（%s）且字符串中有空格，那么字符串将会在第一个空格处被截断（解决方法是使用gets、fgets函数）；
- **【特殊】**特殊情况是%c（字符），由于空格、换行等空白字符本身也是合法字符，所以%c格式标识符不会忽略空白字符，例如下面的例子，要求输入一个字符、一个整数、一个字符：

```
scanf("%c%d%c", &ch1, &a, &ch2);
```

- 读取数值型数据时，若已经读取到了有效数值字符，之后遇到了空白字符、非法字符（例如字母）或已经达到限定宽度，则自动从空白/非法字符/限定宽度处截断，保留前面的有效数值字符。

格式化输入函数: scanf

```c
#include <stdio.h>

int main()
{
    char password[10];
    char c;
    printf("请输入密码: ");
    scanf("%s", password);
    printf("请确认(y/n): ");
    scanf("%c", &c);
    if (c == 'y' || c == 'Y')
        printf("确认成功! ");
}
```



scanf、getchar的错误使用

假设程序要求输入密码，之后要求用户输入y确认密码。左边的代码能够完成上述功能吗?

分析:

- 用户输入密码后，按下回车键完成输入;
- scanf读取字符串abcde，\n仍留在缓冲区内;
- 此时再次调用scanf()，获取了留在缓冲区内的\n，用户将没有机会输入字符y以确认密码。

格式化输入函数: scanf

改进:

```c
#include <stdio.h>

int main()
{
    char password[10];
    char c;
    printf("请输入密码: ");
    scanf("%s", password);
    printf("请确认(y/n): ");
    scanf(" %c", &c);
    if (c == 'y' || c == 'Y')
        printf("确认成功! ");
}
```

方法1：在%c前添加一个空格

```c
#include <stdio.h>

int main()
{
    char password[10];
    char c;
    printf("请输入密码: ");
    scanf("%s", password);
    printf("请确认(y/n): ");
    getchar();
    scanf("%c", &c);
    if (c == 'y' || c == 'Y')
        printf("确认成功! ");
}
```

方法2：添加一句getchar()以清除缓冲区的空白字符

字符串输入/输出函数

gets()

- 作用：读取用户键盘输入的字符串，以回车结束，丢弃换行符，保留剩余字符
- 用法：gets(字符串变量)
- gets允许用户输入含空格的字符串，而scanf("%s", &s)会在空格处截断

puts()

- 作用：在控制台输出字符串，并换行
- 用法：puts(字符串变量)
- puts的作用等价于printf("%s\n", s)

```c
#include <stdio.h>

int main()
{
    char name[50];
    puts("What's your name?");
    gets(name);
    printf("Your name is %s. Hello!", name);
    return 0;
}
```

字符串输入/输出函数

- gets()函数不是内存安全的，因为gets()函数只有一个参数s用于表示字符串存储的地址，而无法判断读取的字符串长度是否超过给s分配的内存大小；
- 可以用fgets()、fputs()替代gets()和puts()。

- **课后练习：自行查阅资料，了解C语言中fgets和fputs函数的用法。**

标准输入输出练习

练习1. 平均分

- 编写一个程序，记录某同学语文、数学、英语三个科目的成绩，并计算三个科目的平均分，平均分**保留两位小数。**

```
请输入语文成绩：90.5
请输入数学成绩：95.5
请输入英语成绩：85
平均成绩：90.33
```

练习2. 温度转换

- 编写一个程序，实现摄氏温度（℃）到华氏温度（°F）的转换，转换结果**保留两位小数。**（备注：摄氏度与华氏度的转换关系为： 华氏度 = 9 / 5 × 摄氏度 +32）

```
请输入摄氏温度：25
华氏温度为：77.00
```

4.2 基本数学函数

数学函数

- **要使用C标准库中的数学函数，需要包含<math.h>头文件**

常用的数学函数包括：

abs()	整数的绝对值
fabs()	浮点数的绝对值
floor()	向下取整
ceil()	向上取整
sin()	正弦函数，输入参数单位为弧度
cos()	余弦函数，输入参数单位为弧度
tan()	正切函数，输入参数单位为弧度
asin()	反正弦函数
acos()	反余弦函数
atan()	反正切函数

sqrt()	平方根
pow()	乘方函数
exp()	e的x次方
pow10()	10的x次方
log()	自然对数（底数是e）
log10()	常用对数（底数是10）

数学函数

- 要使用C标准库中的数学函数，需要包含<math.h>头文件

例1：求直角三角形的斜边

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

int main()
{
    double a, b, h;  // 直角边a、b，斜边h
    printf("请输入直角三角形的两直角边：\n");
    printf("a=");
    scanf("%lf", &a);
    printf("b=");
    scanf("%lf", &b);
    h = sqrt(a * a + b * b);
    printf("斜边h=%.2lf", h);
    return 0;
}
```

例2：三角函数

```c
#define PI 3.1415926

#include <stdio.h>
#include <math.h>

int main()
{
    printf("sin(pi/4)=%lf\n", sin(PI/4));
    printf("cos(pi/4)=%lf\n", cos(PI/4));
    printf("tan(pi/4)=%lf\n", tan(PI/4));
    printf("arcsin(1)=%lf\n", asin(1));
    return 0;
}
```

随机数函数

- **要使用C标准库中的随机数函数，需要包含<stdlib.h>头文件**

C标准库中，用于生成随机数的函数主要包括：

rand()	生成介于0到32767之间的随机整数
srand()	用于为rand()等随机化生成器指定一个种子（seed），如果没有指定，则种子默认为1；

一般把srand和rand函数配合使用，先调用srand函数指定随机数种子，再调用rand函数生成随机数，例如：

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    time_t t;
    srand((unsigned) time(&t));  // 为随机化生成器初始化种子
    printf("%d\n", rand() % 10);  // 随机生成1个随机数，随机数范围是0~9
    return 0;
}
```

思考：如果需要的随机数的范围是0~50呢？10~50呢？

随机数函数

练习：

假设某游戏采用以下方法计算打击伤害：

暴击伤害值=攻击力×(1+暴击伤害)

不暴击伤害值=攻击力

例如，角色的攻击力为1500，暴击率为30%，暴击伤害为150%，那么：

- 角色有30%概率暴击，此时伤害值为：1500×(1+150%)=3750
- 角色有70%概率不暴击，此时伤害值为：1500
- 角色的某次攻击是否暴击是**完全随机的**，不受其他因素影响

试编写一个程序，模拟某一次角色攻击的伤害。

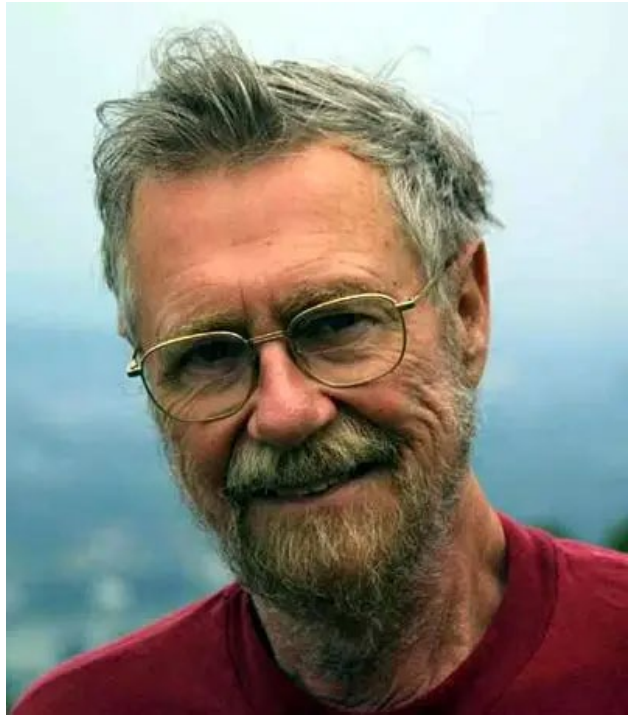
提示：可以使用if判断是否满足某条件

```
if (条件) {
    ...    // 满足条件时执行的语句
} else {
    ...    // 不满足条件时执行的语句
}
```

4.3 结构化程序设计方法

结构化程序设计方法

- **结构化程序设计方法是一种按照模块划分原则提高程序可读性和易维护性、可调性和可扩充性为目标的设计方法。它由荷兰学者E. W. Dijkstra在1965年提出。**
- **结构化程序设计方法的基本结构包括顺序结构、选择结构（分支结构）和循环结构。这些基本结构的共同特点是只允许有一个流动入口和一个出口。**
- **结构化程序设计方法的设计原则包括：**
  - **自顶向下**：从总体目标开始，逐步细化；
  - **逐步求精**：将复杂问题分解为多个子问题；
  - **模块化**：将复杂问题划分为多个功能模块，每个模块独立开发；
  - **限制使用goto语句**：提高程序可读性，避免逻辑混乱。
- **结构化程序设计方法的优点：使程序结构清晰、易于理解和维护，提高软件开发的效率和质量。**

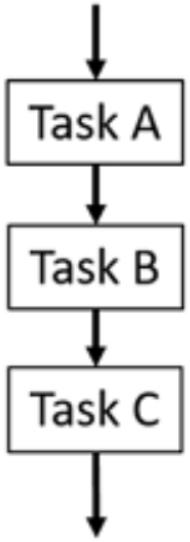


E. W. Dijkstra (1930~2002)

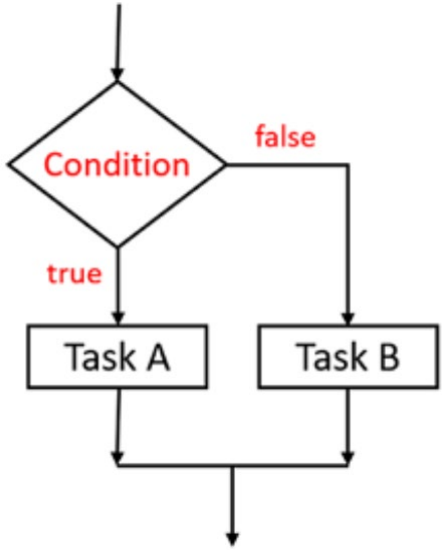
计算机科学领域的重要先驱人物，
ALGOL 60语言的主要开创者，提出
著名的迪克斯特拉最短路算法
1972年图灵奖得主

结构化程序设计方法

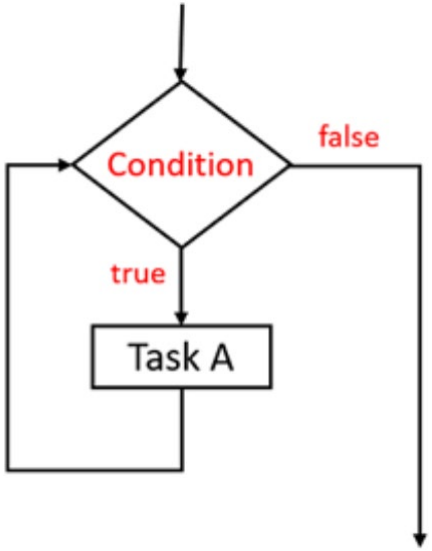
结构化程序设计方法的三种基本结构：



顺序结构
(Sequence)



选择结构
(Selection)

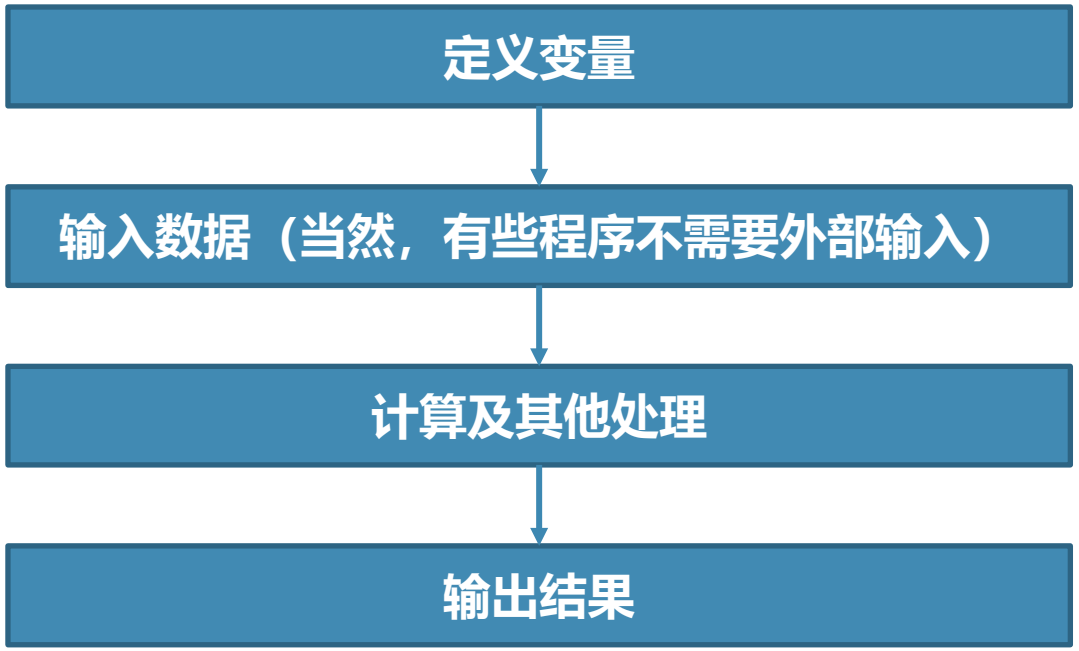


循环结构
(Iteration)

4.4 顺序结构程序设计

顺序结构程序设计

- 顺序结构是最基本的程序结构，代码（指令）从上到下依次执行。
- 典型的顺序结构程序，其框架是：



练习

1. 写一个程序，该程序能够交换两个变量a、b的值。
2. 用户从键盘分别输入两个数字（每个数字在0~9之间），编写程序将这两个数字组成一个两位数。

```
Sample Input: 3  6
Sample Output: 36
```

3. 用户从键盘输入一个三位数，编写程序输出该数字的个位、十位、百位。

```
Sample Input: 123
Sample Output: 1  2  3
```

4. 编写程序，其功能是实现“四舍五入”，保留整数。
5. 编写程序，其功能是实现“四舍五入”，保留两位小数。

练习

1. 写一个程序，该程序能够交换两个变量a、b的值。

```c
#include <stdio.h>

int main(void)
{
    int a = 0, b = 1, temp;
    printf("a=%d, b=%d\n", a, b);
    temp = a;
    a = b;
    b = temp;
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

思考：能否不引入新的变量，实现两个变量a、b的值？（仅考虑a、b是整数的情况）

练习

2. 用户从键盘分别输入两个数字（每个数字在0~9之间），编写程序将这两个数字组成一个两位数。

```c
#include <stdio.h>

int main(void)
{
    int a, b, num;
    scanf("%d%d", &a, &b);
    num = a * 10 + b;
    printf("%d", num);
    return 0;
}
```

练习

3. 用户从键盘输入一个三位数，编写程序输出该数字的个位、十位、百位。

```c
#include <stdio.h>

int main(void)
{
    int num, ones, tens, hundreds;
    scanf("%d", &num);
    hundreds = num / 100;
    tens = (num - hundreds * 100) / 10;
    ones = num - hundreds * 100 - tens * 10;
    printf("%d %d %d", hundreds, tens, ones);
    return 0;
}
```

练习

4. 编写程序，其功能是实现“四舍五入”，保留整数。

```c
#include <stdio.h>

int main(void)
{
    double num;
    int roundNum;
    scanf("%lf", &num);
    roundNum = (int)(num + 0.5);
    printf("%d", roundNum);
    return 0;
}
```

练习

5. 编写程序，其功能是实现“四舍五入”，保留两位小数。

```c
#include <stdio.h>

int main(void)
{
    double num, roundNum;
    scanf("%lf", &num);
    roundNum = (int)(num * 100 + 0.5) / 100.0;
    printf("%.2lf", roundNum);
    return 0;
}
```