

计算机导论与程序设计 [CS006001-60]

段江涛

机电工程学院



2019 年 9 月

lecture-2 算法—程序的灵魂

1 Algorithm + Data Structures = Programs

2 初识 C 语言程序

数据结构与算法

算法 + 数据结构 = 程序

Algorithm + Data Structures =
Programs



沃思 (Niklaus Wirth)

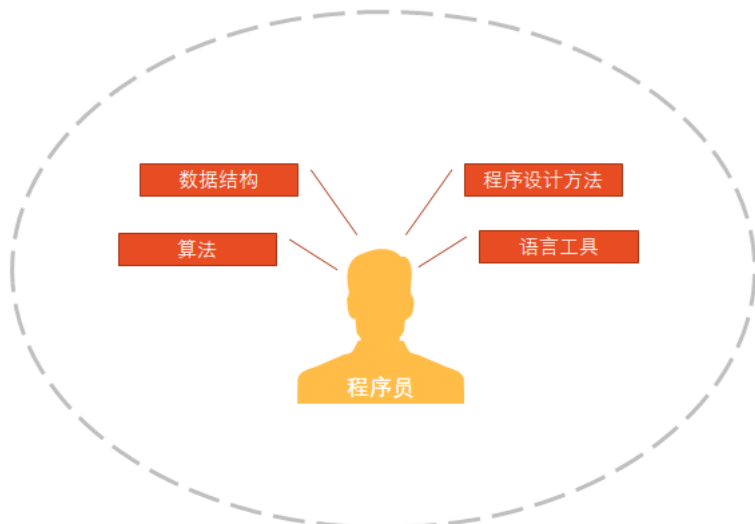
■ 数据结构

对数据的描述。在程序中要指定用到哪些数据,以及这些数据的类型和数据的组织形式。

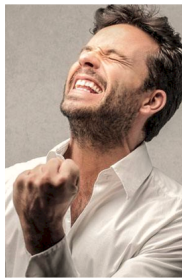
■ 算法

对操作的描述。即要求计算机进行操作的步骤

程序员的工作



算法



算法

- 广义地说,为解决一个问题而采取的方法和步骤,就称为“算法”。
- 对同一个问题,可以有不同的解题方法和步骤。
- 为了有效地进行解题,不仅需要保证算法正确,还要考虑算法的质量,选择合适的算法。

算法

数值运算算法

数值运算的目的是求数值解。

由于数值运算往往有现成的模型，可以运用数值分析方法，因此对数值运算的算法的研究比较深入，算法比较成熟。

非数值运算算法

计算机在非数值运算方面的应用远超在数值运算方面的应用。非数值运算的种类繁多，要求各异，需要使用者参考已有的类似算法，重新设计解决特定问题的专门算法。

简单的算法举例 [例 2.1(p17)]

Example (例 2.1(p17))

求 $1 \times 2 \times 3 \times 4 \times 5$

算法 (一) 步骤

- S1 先求 1 乘以 2, 得到结果 2
- S2 将步骤 1 得到的乘积 2 再乘以 3, 得到结果 6
- S3 将 6 再乘以 4, 得 24
- S4 将 24 再乘以 5, 得 120

思考

求 $1 \times 3 \times 5 \times 7 \times 9$

算法 (二) 步骤

- S1 $p = 1$, 表示将 1 存放在变量 p 中
- S2 $i = 2$, 表示将 2 存放在变量 i 中
- S3 $p = p * i$, 使 p 与 i 相乘, 乘积仍放在变量 p 中
- S4 $i = i + 1$, 使变量 i 的值加 1
- S5 if ($i \leq 5$) goto S3
else 算法结束, 最后得到 p 的值就是 $5!$ 的值。

简单的算法举例 [例 2.2(p18)]

Example (例 2.2(p18))

有 50 个学生, 要求输出成绩在 80 分以上的学生的学号和成绩.

算法步骤

`float g[50]={100,90.5,30.8,...};` \\ 表示 50 名学生成绩

`int i = 0;` \\ 表示第 `i` 个学生学号

`while(i<50)`

{

`printf(" 第%d 个学生成绩%d, ", i+1, g[i]);`

`i = i + 1;`

}

算法 1 例 2.3(p18): 判定 2000—2500 年中的每一年是否为闰年。

```

1: int year=2000, char R;
2: while (year<=2500) do
3:     R='N';
4:     if (year 能被 4 整除, 但是不能被 100 整除) then R='Y';
5:     else if (year 能被 100 整除, 并且能被 400 整除) then
        R='Y';
6:     end if
7:     if (R=='Y') then printf(“%d 是闰年”,year);
8:     else  printf(“%d 不是闰年”,year);
9:     end if
10:    year = year + 1;
11: end while

```



算法 2 例 2.4(p19): 求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{1}{99} - \frac{1}{100}$.

```
1: int sign=1, deno=2;
2: float sum = 1.0;
3: while (deno<=100) do
4:     sign = -1 * sign;
5:     sum = sum + 1/deno;
6: end while
7: printf("sum=%f\n", sum);
```

sign: 表示当前
项的数值符号
deno: 表示当前
项的分母
sum: 表示当前
项的累加和

算法 3 例 2.5(p20): 给出一个大于或等于 3 的正整数, 判断它是不是一个素数.

```
1: int n, i=2, char R='N';
2: scanf("%d",&n); \\ 输入 n 的值;
3: while (i < n) do
4:     if (n 能被 i 整除) then { printf("%d 不是素数", n); return; }
5:     end if
6:     i = i + 1;
7: end while
8: printf("%d 是素数", n);
```

Notes

实际上, n 不必检查被 $2 \sim (n-1)$ 之间的整数整除, 只须检查能否被 $2 \sim \sqrt{2}$ 间的整数整除即可。

算法的特性

1 有穷性

一个算法应包含有限的操作步骤，而不能是无限的

2 确定性

算法中的每一个步骤都应当是确定的，而不应当是含糊的、模棱两可的

3 有零个或多个输入

所谓输入是指在执行算法时需要从外界取得必要的信息

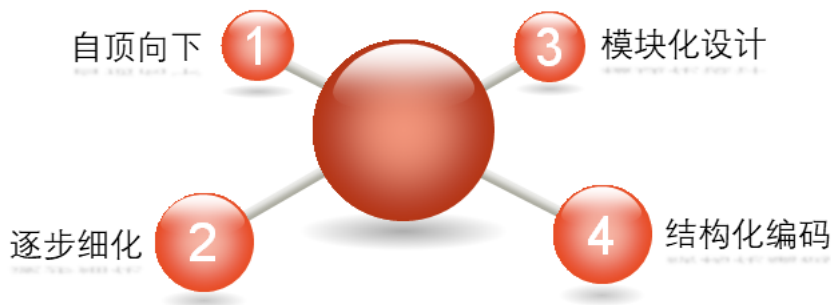
4 有一个或多个输出

算法的目的是为了求解，“解”就是输出

5 有效性

算法中的每一个步骤都应当能有效地执行，并得到确定的结果

结构化程序设计方法



求 5! 的 C 语言程序

```
#include<stdio.h> // standard input/output编译预处理指令

int main() // 主函数
{ // 函数开始标志

    int i,p; // p表示被乘数, i表示乘数

    p=1;
    i=2;
    while(i<=5)
    {
        p=p*i;
        i++; // i = i + 1
    }
    printf("%d\n",p);
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

变量在使用之前首先要定义它的数据类型

```
#include<stdio.h> // standard input/output编译预处理指令

int main() // 主函数
{ // 函数开始标志

    int a,b; // 定义变量a, b为整型数值, 同类型变量可以在一条语句中定义。
    float f; // 定义变量f为单精度浮点数
    double d; // 定义变量d为双精度浮点数
    char c; // 定义变量c为单个英文字母

    a=10;
    b=20;
    f=10.2;
    d=20.3;
    c='A';

    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

输出语句 printf(“原样输出,% 格式符”, 对应变量的值);

```
#include<stdio.h> // standard input/output编译预处理指令

int main() // 主函数
{ // 函数开始标志
    int a=10,b; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
    float f=10.2; // 定义变量f为单精度浮点数
    double d; // 定义变量d为双精度浮点数
    char c; // 定义变量c为单个英文字母
    f=10.2;
    d=20.3;
    c='A';
    printf("a=%d,b=%d,c=%c,f=%f,d=%lf\n",a,b,c,f,d); // \n为换行符
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```


输入语句 scanf(“% 变量格式符”, & 变量名);

```
#include<stdio.h> // standard input/output编译预处理指令

int main() // 主函数
{ // 函数开始标志

    int a=10,b; // 定义变量a, b为整型数值, 定义变量时, 可以指定变量的初值
    float f=10.2; // 定义变量f为单精度浮点数
    double d; // 定义变量d为双精度浮点数
    char c='A'; // 定义变量c为单个英文字母, 字符输入以后讲
    printf("请输入整数a,b,□空格隔开:\n"); // 提示语句[可选]
    scanf("%d%d",&a,&b);

    printf("请输入浮点数f,d,□空格隔开:\n"); // 提示语句[可选]
    scanf("%f%lf",&f,&d);

    printf("a=%d,b=%d,c=%c,f=%f,d=%lf\n",a,b,c,f,d); // \n为换行符
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

if(条件表达式){ 表达式为真 (非 0) 时执行语句; }

```
#include<stdio.h> // standard input/output编译预处理指令

int main() // 主函数
{ // 函数开始标志

    int a=10; // 定义变量a为整型数值，定义变量时，可以指定变量的初值
    if(a>=10)
    {
        printf("a>=10\n"); // \n为换行符
    }
    else
    {
        printf("a<10\n"); // \n为换行符
    }

    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

while(条件表达式){ 表达式为真 (非 0) 时执行语句; }

```
#include<stdio.h> // standard input/output编译预处理指令
int main() // 主函数
{ // 函数开始标志
    int a=10; // 定义变量a为整型数值，定义变量时，可以指定变量的初值
    while (a>=0)
    {
        printf("a=%d\n",a); // \n为换行符
        a--; // a= a - 1
    }
    return 0; // 函数执行完毕返回函数值0
} // 函数结束标志
```

欢迎批评指正！