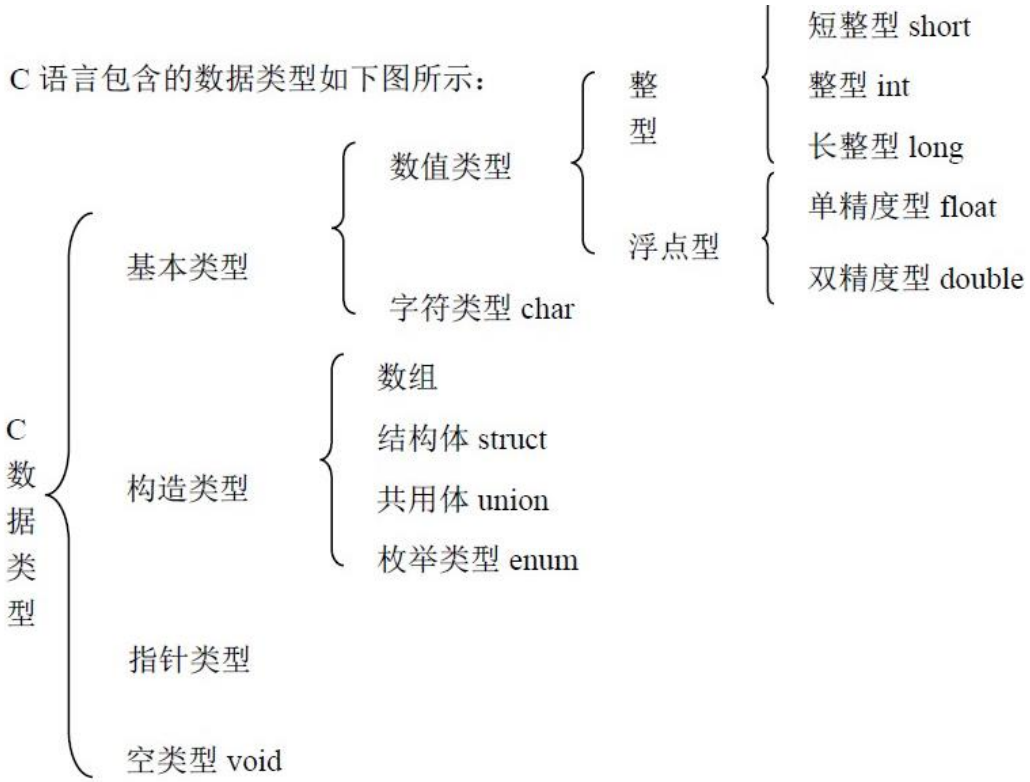
作者&审阅：刘一宇、周梓媛

1. 标识符

标识符是由程序员定义的单词，用来给程序中的数据、函数和其他用户自定义对象命名。它由大写字母 A 到 Z、小写字母 a 到 z、数字 0 到 9 和下划线组成，且第一个字符必须是字母或下划线，随后的字符必须是字母、数字或下划线。且大小写敏感，如 age 和 Age 是两个不同的标识符。

2. 数据类型

☺ **数据类型分类**



☺ **各种类型的存储大小及取值范围**

类型	存储大小	取值范围
char	1 字节	-128 到 127 或 0 到 255
int	2 或 4 字节	-32,768 到 32,767 或 -2,147,483,648 到 2,147,483,647

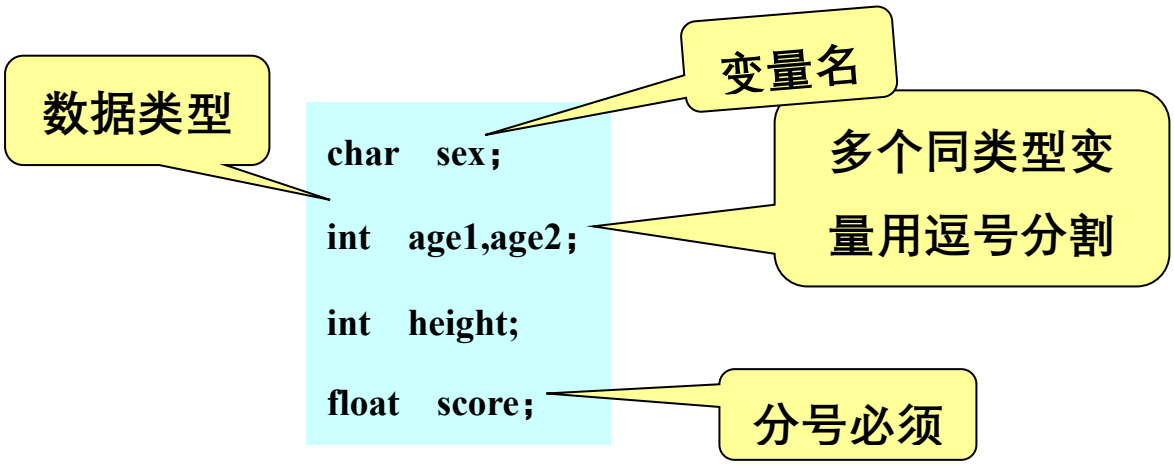
short	2 字节	-32,768 到 32,767
long	4 字节	-2,147,483,648 到 2,147,483,647
float	4 字节	1.2E-38 到 3.4E+38
double	8 字节	2.3E-308 到 1.7E+308

3. 变量

☺ **变量定义**

变量定义就是告诉编译器在何处创建变量的存储，以及如何创建变量的存储。

变量定义指定一个数据类型，并包含了该类型的一个或多个变量的列表



☺ **变量的分类**

局部变量 全局变量

☺ **变量的赋值**

变量赋值：把一个值写入变量代表的存储空间。

C 语言变量赋值格式：变量名=表达式

//先定义变量

```
char    sex;
```

//对变量赋值

```
sex='F';
```

4. 常量

☺ 字面常量

1、整型常量

- 十进制整型常量

符号 ： 0~9　　//例：　1、2、9、111、123

%d ： 格式化输入输出一个十进制数

```
printf ("%d",123);
```

- 八进制整型常量

符号 ： 0~7

前缀 ： 0 用于区分进制 //例：　048 ——八进制

%o ： 格式化输入输出一个八进制整数

```
printf("%o\n",048);
```

- 十六进制整型常量

符号 ： 0~9 ,A~F　A:10　,B:11 ,C:12,… F:15

前缀 ： 0x //例：0x48 ——十进制

%x ： 格式化输入输出一个十六进制整数

```
printf("%x\n",0x48);
```

2、浮点型常量

- 十进制小数

%m.nf ： 格式化输入输出一个浮点型常量

m ： 输出宽度，不足 前补 0，　n ： 小数位数

```
printf("%0.2f",.3.1415);          //保留两位小数输出浮点型常量
```

- 用科学计数法表示浮点型常量

例：3.14 E 2 = 314;　3.14 E-2 = 0.0314;

3、字符型常量

- 可直接打印的字符

用单引号引起来，单引号内不能为空， //%c ： 格式化输入输出字符型常量

例：printf("%c",'a');

printf("%d",'a'); //打印出 'a'的 ASCLL 码值 97

- 转义字符

要使用 '\'

例：printf("%c",'\\'); //打印出 \

printf("%%"); //两个 % 才能打印出一个 %

常见的转义字符：

转义序列	含义
\\	\ 字符
\'	' 字符
\"	" 字符
\?	? 字符
\a	警报铃声
\b	退格键
\f	换页符
\n	换行符
\r	回车
\t	水平制表符
\v	垂直制表符

\ddd	一到三位的八进制数
\xhh	一个或多个数字的十六进制数

2.const 修饰的常变量

```
const int a = 1;
```

const 修饰的常变量，本质上是变量。

3.#define 定义的标识符常量

```
#define PI 3.14159
```

```
#define SIZE 10     //没有等号，最后也没有分号
```

4.枚举常量

5. 输入与输出

☺ printf 输出

1、输出描述性的文字

把输出的文字用双引号包含起来，文字中的\n 表示换行，多个\n 可以换多行。

```
printf("愿我如星君如月，夜夜流光相皎洁。\n");
```

以上代码将在屏幕上输出文字：

愿我如星君如月，夜夜流光相皎洁。

输出文字之后，再输出一个换行。

2、输出整数

输出整数型常量或变量用%d 表示，在参数中列出待输出的整数常量或变量。

```
int age=18;

printf("我年龄是%d 岁。\n", age);
```

3、输出字符

输出字符型常量或变量用%c 表示，在参数中列出待输出的字符常量或变量。

```
printf("我姓别是：%c。\n", 'x');              // 姓别：x-男；y-女

char xb='x';

printf("我姓别是：%c。\n", xb);
```

4、输出浮点数

输出的浮点型常量或变量用%lf 表示，在参数中列出待输出的浮点型常量或变量。

```
printf("我体重是%lf 公斤。\n",62.5);

double weight=62.5;

printf("我体重是%lf 公斤。\n", weight);
```

5、输出字符串

输出字符串常量或变量用%s 表示

☺ scanf 输入

scanf 函数是格式化输入函数，用于接受从键盘输入的数据，用户输入数据完成后，按回车键（Enter）结束输入。

1、输入整数

```
int age=0;
```

```
scanf("%d", &age);
```

2、输入字符

```
char xb=0;

scanf("%c", &xb);
```

3、输入浮点数

```
double weight=62.5;

scanf("%lf", &weight);
```

4、输入字符串

```
 char name[20];

scanf("%s", name);     //注意这里没有&哦
```

6. 运算符

☺ 算术运算符

假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

运算符	描述	实例
+	把两个操作数相加	A + B 将得到 30
-	从第一个操作数中减去第二个操作数	A - B 将得到 -10
*	把两个操作数相乘	A * B 将得到 200
/	分子除以分母	B / A 将得到 2
%	取模运算符，整除后的余数	B % A 将得到 0

++	自增运算符，整数值增加 1	A++ 将得到 11
--	自减运算符，整数值减少 1	A-- 将得到 9

☺ 关系运算符

假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

运算符	描述	实例
==	检查两个操作数的值是否相等，如果相等则条件为真。	(A == B) 为假。
!=	检查两个操作数的值是否相等，如果不相等则条件为真。	(A != B) 为真。
>	检查左操作数的值是否大于右操作数的值，如果是则条件为真。	(A > B) 为假。
<	检查左操作数的值是否小于右操作数的值，如果是则条件为真。	(A < B) 为真。
>=	检查左操作数的值是否大于或等于右操作数的值，如果是则条件为真。	(A >= B) 为假。
<=	检查左操作数的值是否小于或等于右操作数的值，如果是则条件为真。	(A <= B) 为真。

☺ 逻辑运算符

假设变量 **A** 的值为 1，变量 **B** 的值为 0，则：

运算符	描述	实例
&&	称为逻辑与运算符。如果两个操作数都非零，则条件为真。	(A && B) 为假。
||	称为逻辑或运算符。如果两个操作数中有任意一个非零，则条件为真。	(A || B) 为真。
!	称为逻辑非运算符。用来逆转操作数的逻辑状态。如果条件为真则逻辑非运算符将使其为假。	!(A && B) 为真。

☺ 位运算符

位运算符作用于位，并逐位执行操作。&、 | 和 ^ 的真值表如下所示：

p	q	p & q	p | q	p ^ q

0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

☺ 赋值运算符

运算符	描述	实例
=	简单的赋值运算符，把右边操作数的值赋给左边操作数	C = A + B 将把 A + B 的值赋给 C
+=	加且赋值运算符，把右边操作数加上左边操作数的结果赋值给左边操作数	C += A 相当于 C = C + A
-=	减且赋值运算符，把左边操作数减去右边操作数的结果赋值给左边操作数	C -= A 相当于 C = C - A
*=	乘且赋值运算符，把右边操作数乘以左边操作数的结果赋值给左边操作数	C *= A 相当于 C = C * A
/=	除且赋值运算符，把左边操作数除以右边操作数的结果赋值给左边操作数	C /= A 相当于 C = C / A
%=	求模且赋值运算符，求两个操作数的模赋值给左边操作数	C %= A 相当于 C = C % A
<<=	左移且赋值运算符	C <<= 2 等同于 C = C << 2
>>=	右移且赋值运算符	C >>= 2 等同于 C = C >> 2
&=	按位与且赋值运算符	C &= 2 等同于 C = C & 2
^=	按位异或且赋值运算符	C ^= 2 等同于 C = C ^ 2
|=	按位或且赋值运算符	C |= 2 等同于 C = C | 2

☺ 杂项运算符 ↦ sizeof　&　三目运算符

运算符	描述	实例
sizeof()	返回变量的大小。	sizeof(a) 将返回 4，其中 a 是整数。
&	返回变量的地址。	&a; 将给出变量的实际地址。
*	指向一个变量。	*a; 将指向一个变量。
? :	条件表达式	如果条件为真 ? 则值为 X : 否则值为 Y

☺ 运算符优先级

类别	运算符	结合性
后缀	() [] -> . ++ - -	从左到右
一元	+ - ! ~ ++ - - (type)* & sizeof	从右到左
乘除	* / %	从左到右
加减	+ -	从左到右
移位	<< >>	从左到右
关系	< <= > >=	从左到右
相等	== !=	从左到右
位与 AND	&	从左到右
位异或 XOR	^	从左到右
位或 OR	|	从左到右
逻辑与 AND	&&	从左到右

逻辑或 OR	||	从左到右
条件	?:	从右到左
赋值	= += -= *= /= %=>>= <<= &= ^= |=	从右到左
逗号	,	从左到右

3.2 语句 （statement）

①表达式语句

在**表达式**后加上分号**";"**（分号是 C 语言中语句的结束符）

如果表达式涉及到赋值（存在**赋值运算符**或运算符**++**或**--**，如 **i = i + 10 * j;** ），则将计算得到的值保存到变量中。

如果不涉及赋值（如语句 **i<j;**），则值将被丢弃.

②复合语句

复合语句是包含**零个或多个**语句的代码单元，使得一组语句成为一个整体，也被称为**块**。在 C 语言中，复合语句由一个左大括号、可选语句段、一个右大括号组成。如：

```
{
  i = 1;
  j = 2 * i;
}
```

③选择语句

- if-else 语句

```
if (表达式)
        语句 1
else
        语句 2
```

表达式必须用括号括起来

可以是任意语句，多个语句要加{ }，语句 2 同理

else 语句不可单独使用，必须与 if 配对，但 if 可以没有 else 子句

- switch-case 语句

```
switch (表达式)
{
    case 常量整数型表达式 1:
        语句 1;break;
    case 常量整数型表达式 2:
        语句 2;break;
          .....
    case 常量整数型表达式 n:
        语句 n;break;
    default:
        语句 n + 1
```

break:跳出 switch 语句

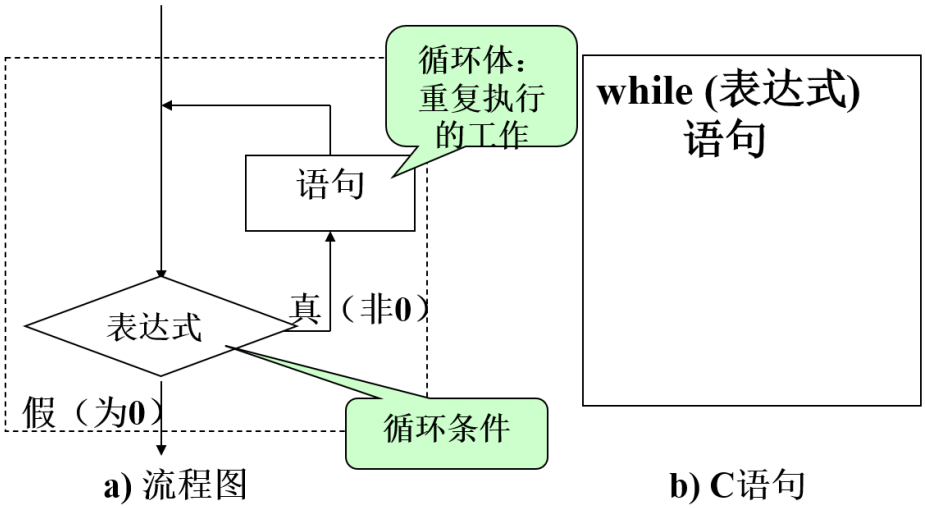
如果没有任何一个 case 后面的"常量整数型表达式"的值与"表达式"的值匹配，则执行 default 后面的语句（组）

}

- if 语句的嵌套

④循环结构语句

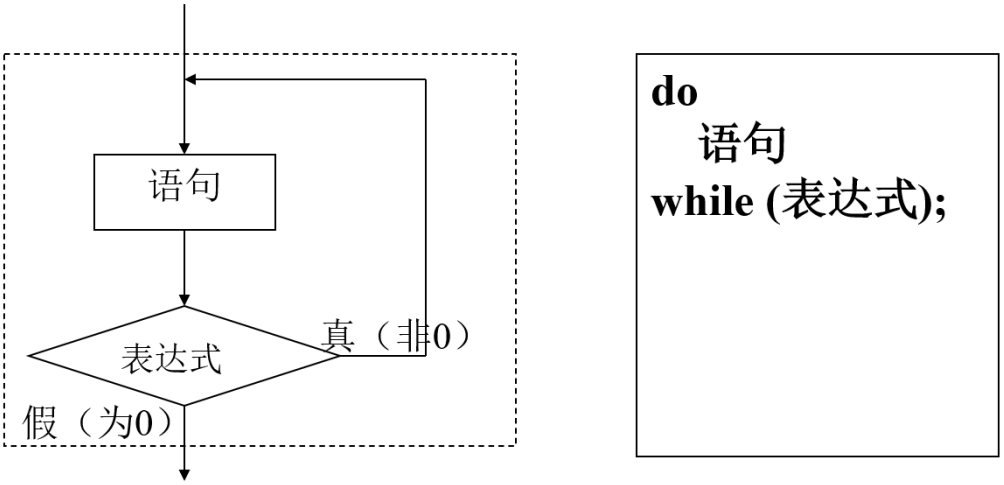
- while



a) 流程图　　b) C语句

可以看出语句（组）可能不会执行（第一次判断表达式结果就为假）。

- do-while



a）流程图　　b) C语句

可以看出语句（组）至少会执行一次。

- for

for (表达式 1；表达式 2；表达式 3)
　　语句

表达式 1、3 根据需要可有可无

等价于

表达式 1；//循环初始化
While（表达式 2）
{
　　语句
表达式 3;

}

3.3 部分 C 语言语句

- break 和 continue 语句

  break：强行结束全部循环（跳出循环，转向执行循环语句后的语句）
  continue：强行结束当前循环（结束第 n 次循环进入第 n+1 次循环）

- 一些常用的转义序列

转义序列	含 义 描 述
\n	换行。将光标定位到下一行的开始位置。
\t	水平制表符。把光标跳到tab键的下一个输出区。
\r	回车。把光标定位在当前行（而不是下一行）的开始位置。
\a	响铃。使系统铃发声。
\b	光标回退一个字符。
\\	反斜扛。打印一个反斜扛字符。
\"	双引号。打印一个双引号字符。

- 再论 C 语言中的数据类型

(1) short、long 限定整数类型，short int、 long int、long long int 可分别简写为 short，long 和 long long。
(2) long 可限定 double 类型
(3) signed、unsigned 限定 char 类型和任何整数类型
  - 经 unsigned 限定的类型取值范围必须是正的或者为 0。一般省略 signed，如 signed char 通常写成 char。
  - signed char：－128～127（1 字节）
  - unsigned char：0～255 （1 字节）

(4) 转换说明符

short：%hd,　　unsigned short ： %hu
int ： %d,　　unsigned int ： %u
long ： %ld,　　unsigned long ： %lu
long long：%lld　　unsigned long long：%llu

- 混合运算时各种数据类型之间的转换

(1)

	数据类型	scanf函数的转换说明符	printf函数的转换说明符
高	long double	%Lf	%Lf
	double	%lf	%lf
	float	%f	%f
	unsigned long	%lu	%lu
	long	%ld	%ld
	unsigned int	%u	%u
	int	%d	%d
	unsigned short	%hu或%u	%hu或%u
	short	%hd	%hd
	unsigned char	%u	%u
低	char	%c	%c

（2）在一个赋值语句中，如果赋值运算符左侧变量的类型和右侧表达式的类型不一致，则赋值时将进行自动类型转换，将右侧表达式的值转换成左侧变量的类型。

（3）类型转换运算符：(类型说明符)表达式 或者 (类型说明符)(表达式)，用于将表达式的运算结果类型转换为类型说明符指定的数据类型。

➢ 自增和自减运算符

运算符	表达式范例	功能描述
++	++a	先将 a 加 1,然后把 a 的新值用在出现变量 a 的表达式中
++	a++	在出现变量 a 的表达式中使用 a 的当前值,然后将 a 加 1
--	--a	先将 b 减 1,然后把 b 的新值用在出现变量 b 的表达式中
--	a--	在出现变量 b 的表达式中使用 b 的当前值,然后将 b 减 1

ANSI 没有规定运算符操作数的计算顺序，因此如果在一条语句中将特定变量自增或自减不止一次时，程序员应该尽量避免使用自增或自减运算符。

➢ 条件运算符和逗号运算符

条件运算符 **表达式 1 ？表达式 2 ：表达式 3**（C 语言唯一的三目运算符）

```
ch=((ch>='A' && ch<='Z') ? (ch+32) : ch);
```

↕

```
if (ch>='A' && ch<='Z')
ch= ch+32;
```

➢ 运算符的优先级和结合性

运算符		类型
( )		圆括号
++ -- - (类型) !		单目运算符
* / %		乘除法运算符
+ -		加法运算符
< <= > >=		关系运算符
== !=		相等测试运算符
&&		逻辑“与”
||		逻辑“或”
?:		条件运算符
= += -= *= /= %=		赋值运算符
,		逗号运算符

高

低

运算符优先级

单目运算符高于算术运算符高于关系运算符高于逻辑运算符高于条件运算符高于赋值运算符高于逗号运算符

运算符	结合性	类型
( )	自左向右	圆括号
++ -- - (类型) !	自右向左	单目运算符
* / %	自左向右	乘除法运算符
+ -	自左向右	加法运算符
< <= > >=	自左向右	关系运算符
== !=	自左向右	相等测试运算符
&&	自左向右	逻辑“与”
||	自左向右	逻辑“或”
?:	自右向左	条件运算符
= += -= *= /= %=	自右向左	赋值运算符
,	自左向右	逗号运算符

运算符结合性

4.1 算法的概念

算法：是解决问题的步骤序列（操作序列）。

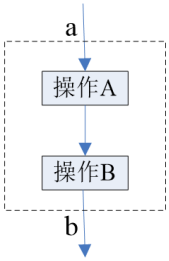
算法必须具备的五个特性：

1. 可执行性：算法中的每一个步骤都是计算机可执行的（在计算机能力集范围内）
2. 确定性：算法中的每一个步骤，必须是明确定义的，不得有任何歧义性

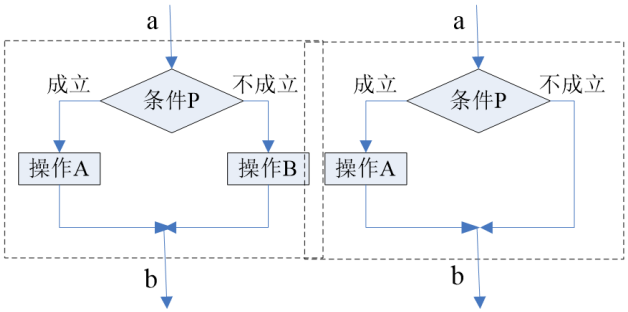
3. 有穷性：算法必须在执行有穷步之后结束
4. 有输入信息的说明：对加工对象提要求
5. 有输出信息的步骤：至少要输出问题答案

4.2 算法的三种基本结构

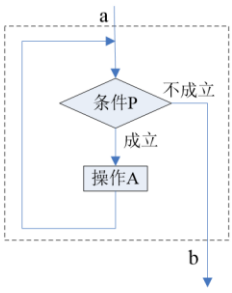
①顺序结构



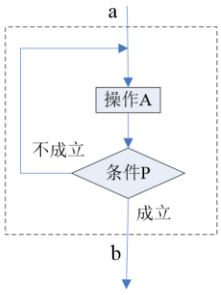
②选择结构



③循环结构



当型循环结构



直到型循环结构

三种结构特点：

- **只有一个入口（a 处）**
- **只有一个出口（b 处）**
- **结构内的每一个部分都有机会被执行到**
- **结构内不存在“死循环”（无终止的循环）**

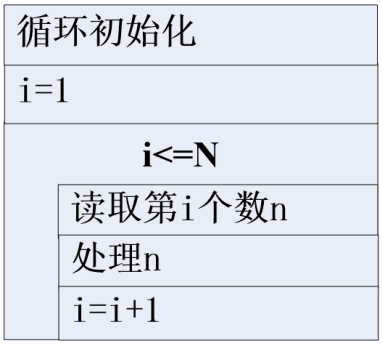
4.3 迭代算法与循环结构

①迭代算法解题

确定迭代变量 a_i→建立迭代关系式→对迭代过程进行控制

②循环次数已知/未知的处理模式

计数器控制的循环



标记控制的循环

4.4 算法的描述方法

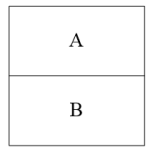
- 用自然语言描述

  文字冗长；不严格，易产生歧义（二义性）；不方便描述分支和循环结构。

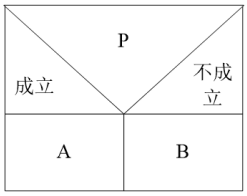
- 用流程图描述

  对流程线的使用没有严格限制，阅读困难；不能保证算法结构的单入单出特性；占用篇幅较多

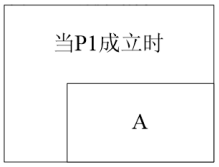
- 用 N-S 流程图描述



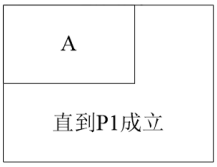
顺序结构



选择结构



当型循环结构



直到型循环结构

- **适合于表示算法，而在算法设计中使用不是很理想。**

- 用伪码描述

  用介于自然语言和程序设计语言之间的文字和符号来描述算法。

- 用计算机语言描述

4.5 结构化程序设计方法

结构化程序设计方法的基本思想：采用分而治之的方法，将一个复杂问题分解为相对

简单的一些子问题，然后针对这些子问题进行求解。如果某个子问题仍然是比较复杂的，再进一步分解为子-子问题，直到所有问题都能够求解。

结构化程序设计方法：

自顶向下；
逐步细化；
模块化设计（函数）；
结构化编码（三种基本结构）。

4.6 穷举算法

穷举法解题思路：

① 明确所有的组合情况；
② 检查每一种组合情况是否满足条件。

5.1 子程序设计

- 子程序是封装并给以命名的一段程序代码，这段程序代码完成子程序所定义的功能，可供调用。
- 子程序很重要的特点：调用者只需要关心子程序接口，不必了解子程序内部实现细节。
- 子程序的控制和调用机制
  ①通过子程序名进行调用；
  ②调用时需要传递一些供子程序计算和处理的数据（参数）；
  ③子程序执行完成后需要返回处理结果；
  ④子程序可以调用其他的子程序；