# 2

# 程序设计入门

- 从问题中抽象出计算机能存储和处理的数据
- 根据数据的特性确定计算机能执行的运算规则
- 针对问题的特征设计能控制计算过程的算法
- 在计算机内部以及与外界之间传输数据

## 数据与数据类型

**例 2.1**

```
r = 2.0;
s = 3.14159 * r * r ;  // s = PI * r * r ;
```

常量：程序运行期间，其值不能或不允许被改变的数据对象

变量：程序运行期间，其值可以改变的数据对象

（变量有名字、值和类型三个属性）

## 数据与数据类型

**标识符（identifier）**

- 用来标识变量、常量、函数等的字符序列

**组成：**

- 只能由字母、数字、下划线组成，且第一个字母必须是字母或下划线
- 大小写敏感
- 不能使用关键字

**命名原则：**

- 见名知意
- 不宜混淆 如l与I，o与0

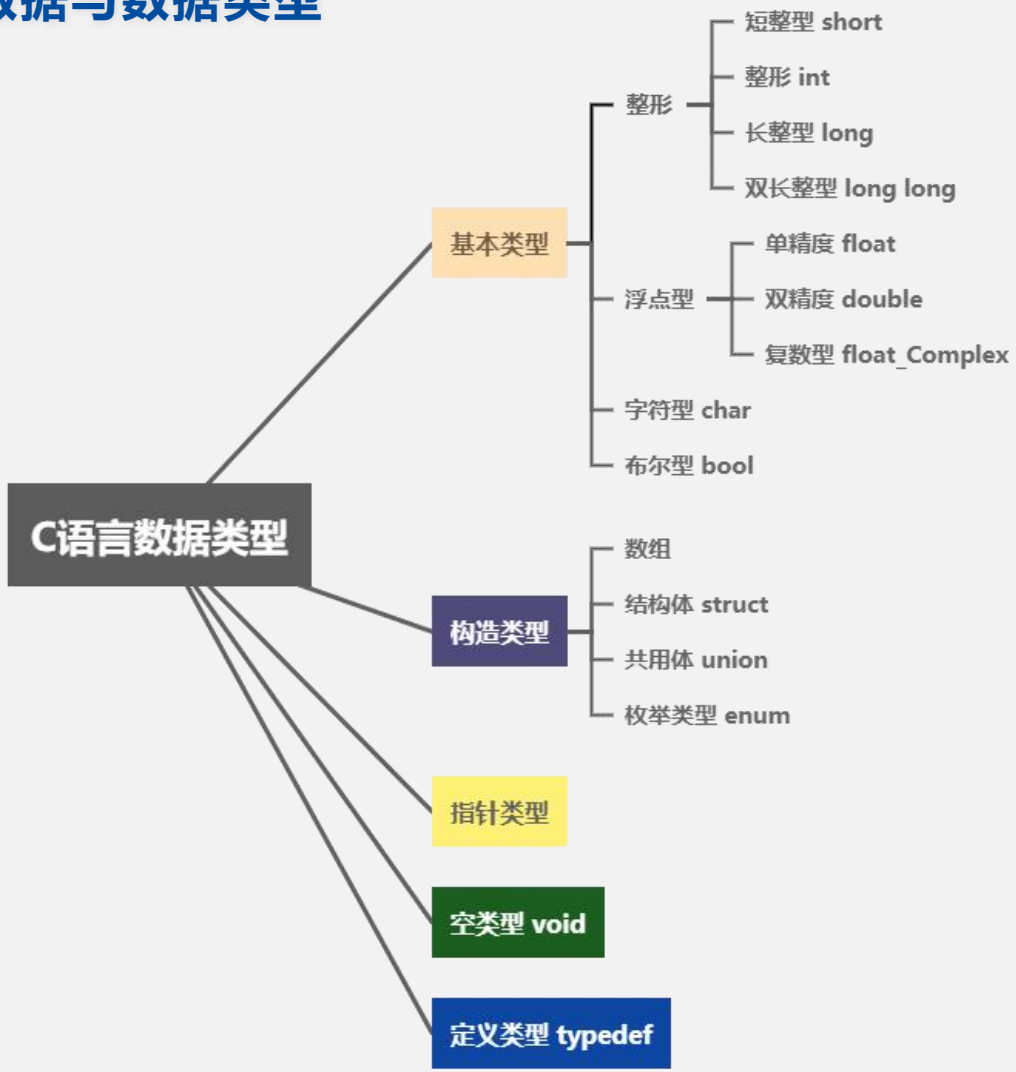
**例 2.2** 判断下列标识符号合法性

sum Sum M.D.John day Date 3days

student_name #33 lotus_1_2_3

char a>b _above $123

# 数据与数据类型




数据类型决定：

1. 数据占内存字节数
2. 数据取值范围
3. 其上可进行的操作

## 数据与数据类型

| 类型 | 符号 | 关键字 | 所占位数 | 数的表示范围 |
|---|---|---|---|---|
| 整型 | 有 | (signed) int | 32 | -2147483648 - 2147483647 |
| | | (signed) short | 16 | -32768 - 32767 |
| | | (signed) long | 32 | -2147483648 - 2147483647 |
| | 无 | unsigned int | 32 | 0 - 4294967295 |
| | | unsigned short | 16 | 0 - 65535 |
| | | unsigned long | 32 | 0 - 4294967295 |
| 实型 | 有 | float | 32 | 4.4e-384.4e38 |
| | 有 | double | 64 | 1.7e~308 - 1.7e~308 |
| 字符型 | 有 | char | 8 | -128 - 127 |
| | 无 | unsigned char | 8 | 0 - 255 |

**说明:** 数据类型所占字节数可能随系统环境不同而不同

## 数据与数据类型

变量定义的一般格式：数据类型 变量1[，变量2，…，变量n];

决定分配字节数和数的表示范围

合法标识符

**例 2.2**

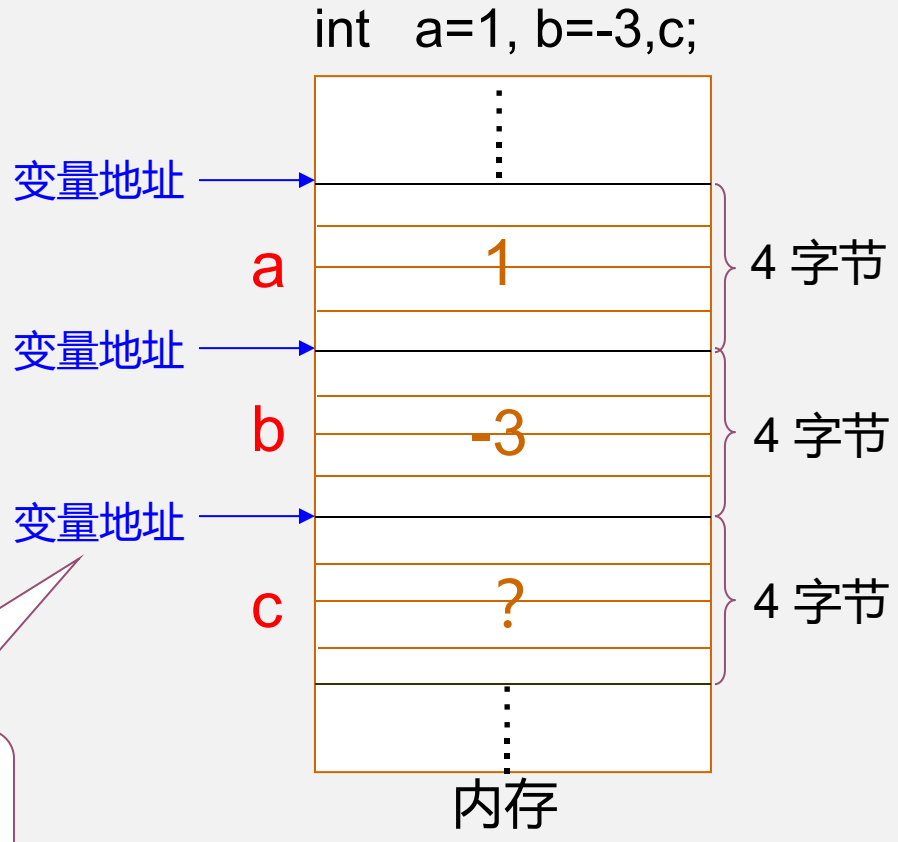
```
int  a,b,c;
float  data;
```

## 数据与数据类型

变量初始化:定义时赋初值

**例 2.3**

```
int  a=1,b=-3,c;
float  data=4.67;
char  ch='A';
int  x=1,y=1,z=1;
int  x=y=z=1;
```



## 数据与数据类型

变量：先定义，后使用

**例 2.4**

```
int   student;
stadent=19;    //Undefined symbol ‘statent’ in function main
```

**例 2.5**

```
float  a,b,c;
c=a%b;    // illegal  use of floating point  in function main
```

## 数据与数据类型

变量定义位置：一般放在函数开头

**例 2.6**

```
main()
{   int  a,b=2;
    a=1;
    float  data;
    data=(a+b)*1.2;
    printf(“data=%f\n”,data);
}
```

## 运算符与表达式

```
r = 2.0;
s = 3.14159 * r * r ;
```

- **操作数**：常量和变量都是操作数，也称为操作对象或运算对象；
- **运算符**："*"和"="都是运算符；
- **算术运算**："*"是乘法运算符，有浮点数参与的运算，规则与数学运算一样，得到一个浮点数结果；
- **表达式**：运算符与操作数的组合称为表达式
- **赋值运算**："="不是数学上的相等符号，而是赋值运算符

## 运算符与表达式

```
r = 2.0;
s = 3.14159 * r * r ;
```

- **优先级**：表达式中包含多个运算符时，按运算符的优先级确定计算顺序
- **结合性**：一个操作数两边都有运算符，且优先级相同时，按运算符的结合性确定计算顺序，结合性分为左结合与右结合两种，大部分二元运算符是左结合的
- **运算次序**：最终的运算次序是先算左边的"3.14159 * r"，再算右边的"* r"，最后算前边的"s="
- **类型转换**：当二元运算符两边的操作数类型不同时，会自动（隐式）进行类型转换
- **语句**：在表达式后面添加英文字符";"，就构成了一条可执行的 C 程序语句。可执行语句可以认为是最基本、最简单的“自动化”。

## 运算符与表达式

**C语言常用运算符**

| | |
|---|---|
| 算术运算符: | + - * / % ++ -- |
| 关系运算符: | < <= == > >= != |
| 逻辑运算符: | ! && \|\| |
| 位运算符 : | << >> ~ \| ^ & |
| 赋值运算符: | = 及其扩展 |
| 条件运算符: | ? : |
| 逗号运算符: | , |
| 指针运算符: | * & |
| 求字节数 : | sizeof |
| 强制类型转换: | (类型) |
| 分量运算符: | . -> |
| 下标运算符: | [ ] |
| 其它 : | ( ) - |

## 2.3 输出与输入

### 用printf函数输出

**格式:**

printf( " …… " , 输出项1, 输出项2 , …… , 输出项n );

格式控制
(转换控制字符串)

输出表列
(用逗号分隔的数据组,可选项)

**例 2.7** printf( “x=%d,y=%f\n” , x , x+3 );

## 用printf函数输出

| | | | |
|---|---|---|---|
| d,i | 十进制整数 | int a=567;printf ( “%d”,a); | 567 |
| x,X | 十六进制无符号整数 | int a=255;printf(“%x”,a); | ff |
| o | 八进制无符号整数 | int a=65;printf(“%o”,a); | 101 |
| u | 不带符号十进制整数 | int a=567;printf(“%u”,a); | 567 |
| c | 单一字符 | char a=65;printf(“%c”,a); | A |
| s | 字符串 | printf(“%s”,“ABC”); | ABC |
| e,E | 指数形式浮点小数 | float a=567.789;printf(“%e”,a); | 5.677890e+02 |
| f | 小数形式浮点小数 | float a=567.789;printf(“%f”,a); | 567.789000 |
| g | e和f中较短一种 | float a=567.789;printf(“%g”,a); | 567.789 |
| %% | 百分号本身 | printf(“%%”); | % |

- 格式字符要用小写
- 格式字符与输出项个数应相同，按先后顺序 一 一 对应
- 输出转换:格式字符与输出项类型不一致,自动按指定格式输出

## C程序与函数

### 例 2.8

```
#include<stdio.h>          // C 程序最常见的第一行，称为“文件包含”，内有函数的说明
int main()                 //C 程序必须有一个 main 函数，也总是从 main 函数开始执行
{                          //大括号中包含的是 main 函数的函数体
    float bmi;             //声明存储 BMI 值的浮点型变量 bmi，其中的值未定
    bmi=80/1.85/1.85;      //计算与赋值语句，结果存储在 bmi 中
    printf("%f",bmi);      //输出语句，用十进制小数格式在屏幕上打印变量 bmi 的值
    return 0;              //程序执行完毕返回到操作系统
}
```

## C程序与函数

### 例 2.8

```
#include<stdio.h>
int main()
{
    float bmi;
    bmi=80/1.85/1.85;
    printf("%f",bmi);
    return 0;
}
```

文件包含

函数 函数类型

返回语句

## C程序与函数

### 例 2.8

```
#include<stdio.h>          // C 程序最常见的第一行，称为“文件包含”，内有函数的说明
int main()                 //C 程序必须有一个 main 函数，也总是从 main 函数开始执行
{                          //大括号中包含的是 main 函数的函数体
    float bmi;             //声明存储 BMI 值的浮点型变量 bmi，其中的值未定
    bmi=80/1.85/1.85;      //计算与赋值语句，结果存储在 bmi 中
    printf("%f",bmi);      //输出语句，用十进制小数格式在屏幕上打印变量 bmi 的值
    return 0;              //程序执行完毕返回到操作系统
}
```

## 用scanf函数输入

**格式：**

scanf( " …… " , &变量名1 , &变量名2 , …… , &变量名n );

格式控制
(转换控制字符串)

地址表列
(简单变量要用取地址运算符&)

**例 2.9** scanf( “%d,%c” , &age , &sex );

%c： 接收任意的单个字符并转换为 ASCII 码整数

%d： 接收一串连续的数字字符并转换为一个整数

%f： 接收一串可包含小数点的连续的数字字符并转换为一个单精度浮点数

%lf： 接收一串可包含小数点的连续的数字字符并转换为一个双精度浮点数

## C程序与函数

### 例 2.10

```
#include<stdio.h>                    // scanf 的说明也在 stdio.h 中
int main()
{
        float bmi,height;
        int weight;
        scanf("%d",&weight);         //从键盘接收整数格式的体重值
        scanf("%f",&height);         //从键盘接收小数格式的身高值
        bmi=weight/height/height;    //计算 bmi 的语句
        printf("%f",bmi);            //在屏幕上打印 bmi 值
        return 0;
}
```

## 2.4 条件判断与选择

**例 2.11**

```
if(bmi>25)
        printf("你超重了哦！");
else
        printf("你没有超重。");
```

## 关系运算

关系运算符：< <= == >= > !=

结合方向：自左向右

关系运算符的结果只有两种： 成立（真） 1

不成立（假） 0

**例 2.12** int a=3,b=2,c=1,d,f;

| | |
|---|---|
| a>b | 1 |
| (a>b)==c | 1 |
| b+c<a | 0 |
| d=a>b | 1 |
| f=a>b>c | 0 |

## if-else语句与流程图

简单if语句

格式： if（表达式）

语句1/复合语句1

选择if语句

格式： if（表达式）

语句1/复合语句1

else

语句2/复合语句2

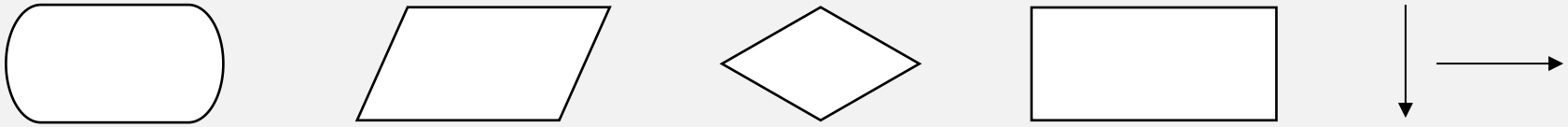
**例 2.12**

```
#include<stdio.h>
int main()
{
    float bmi,height;
    int weight;
    printf("输入体重（千克）：");
    scanf("%d",&weight);
    printf("输入身高（米）：");
    scanf("%f",&height);
    bmi=weight/height/height;
    if(bmi>25)
            printf("你超重了哦！");
    else
            printf("你没有超重。");
    return 0;
}
```

## if-else语句与流程图

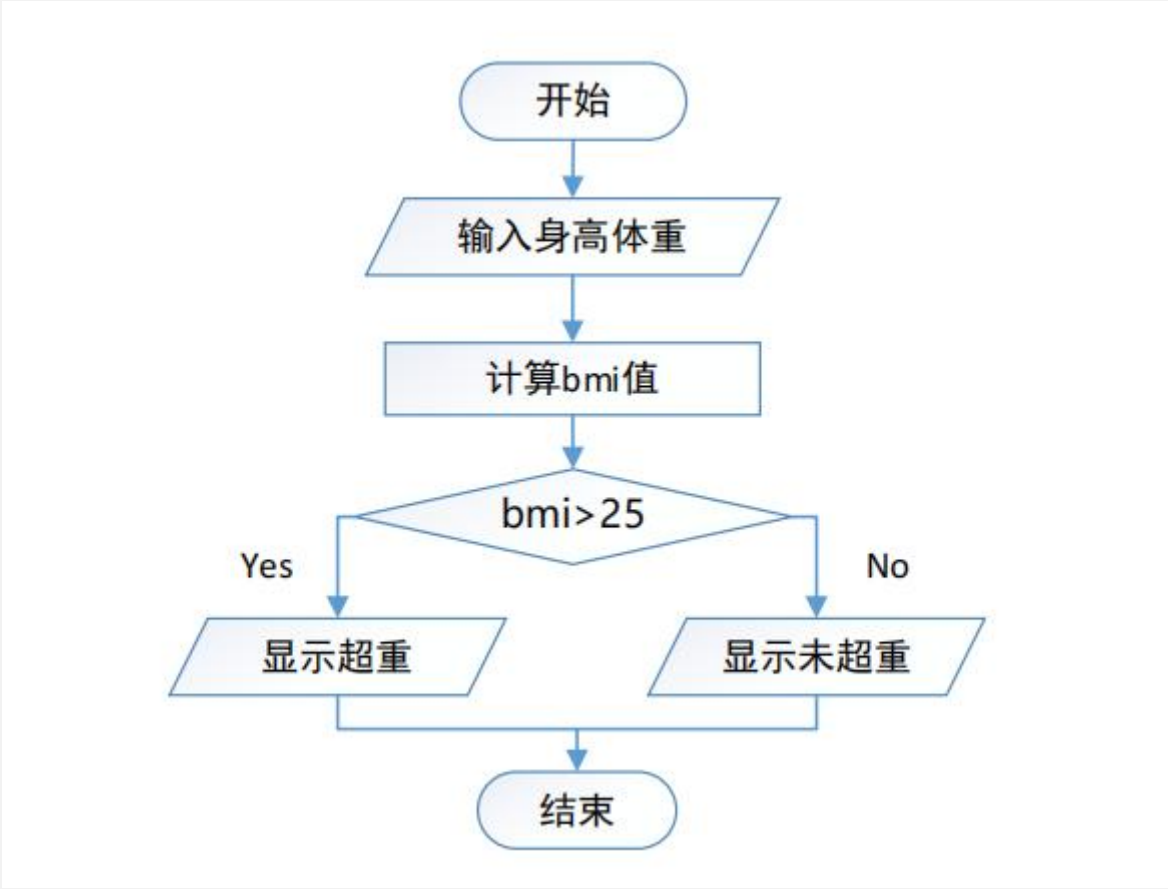
**流程图**：用一些约定的几何图形来描述算法。用某种图框表示某种操作，用箭头表示算法流程。

流程图（的符号及意义）美国标准化协会ANSI规定了一些常用的流程图符号，已为世界各国程序工作者普遍采用。



起止框　输入输出框　判断选择框　处理框　流程线

## if-else语句与流程图



## 逻辑运算

逻辑运算符： &&  ||  !

结合方向：自左向右（逻辑与、逻辑或）

| a | b | !a | !b | a&&b | a\|\|b |
|---|---|---|---|---|---|
| 真 | 真 | 假 | 假 | 真 | 真 |
| 真 | 假 | 假 | 真 | 假 | 真 |
| 假 | 真 | 真 | 假 | 假 | 真 |
| 假 | 假 | 真 | 真 | 假 | 假 |

## 逻辑运算

逻辑运算符： && || !

结合方向：自左向右（逻辑与、逻辑或）

**例 2.13**

```
if( bmi>=20&&bmi<=25 )
        printf("你的体重很标准。");
else
        printf("你的体重超标了哦。");
```

## 逻辑运算

- 短路特性：逻辑表达式求解时，并非所有的逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符

```
例  a&&b&&c        //只在a为真时，才判别b的值；
                     只在a、b都为真时，才判别c的值
例  a||b||c        //只在a为假时，才判别b的值；
                     只在a、b都为假时，才判别c的值
例  a=1;b=2;c=3;d=4;m=1;n=1;
    (m=a>b)&&(n=c>d)      //结果m=0,n=1
```

## 2.5 循环与数组

### for循环与伪代码

for语句一般应用形式：

```
for(表达式1；表达式2；表达式3)
{
        循环体语句组4;
}
```

**例 2.14** 用for循环求 $\sum_{n=1}^{100} n$

```
#include <stdio.h>
main()
{   int i,sum=0;
    for(i=1;i<=100;i=i+1)
        sum+=i;
    printf("%d",sum);
}
```

- 习惯性用法表达式1为循环变量赋初值，表达式2为循环条件，表达式3为循环变量增值；
- for语句中expr1, expr2 ,expr3 类型任意，都可省略，但分号；不可省；
- for语句可以转换成while结构。

## for循环与伪代码

**例 2.15**

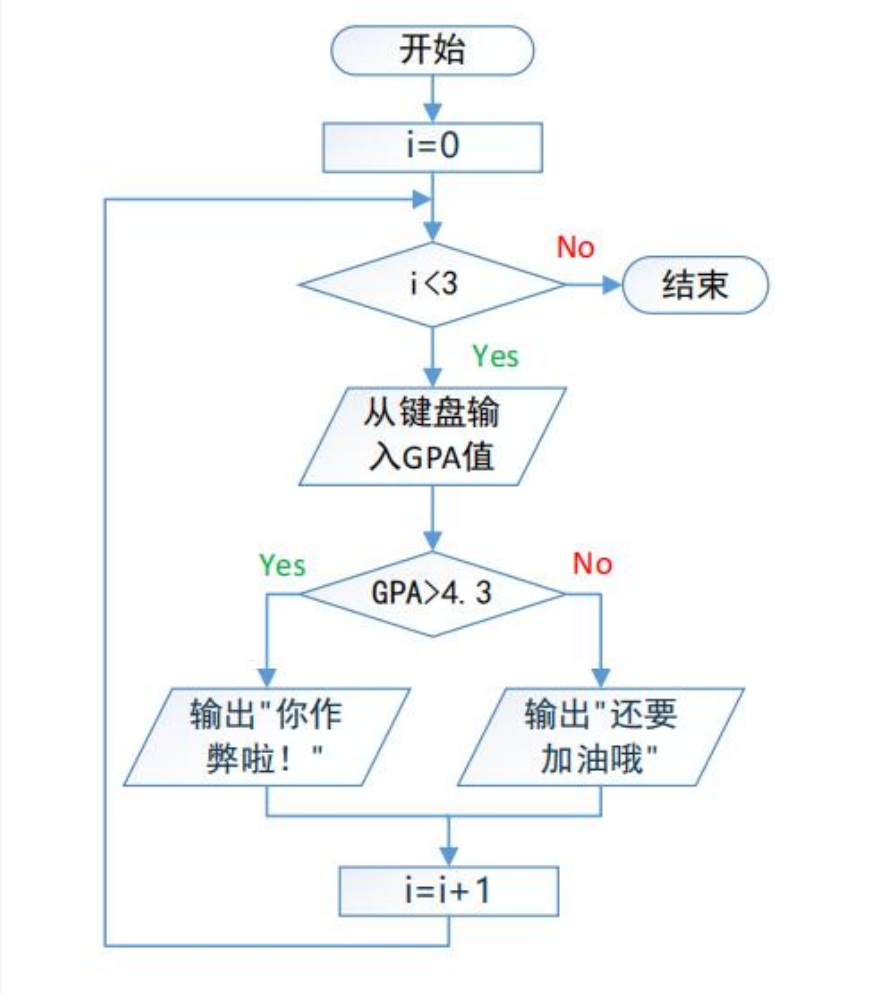
```
#include<stdio.h>
int main()
{
	float GPA;
	int i;
	for(i=0 ; i<3 ; i++)
	{
		scanf(" %f ", &GPA);
		if( GPA > 4.3 )
			printf( "你作弊啦！ " );
		else
			printf( "还要加油哦~" );
	}
	return 0;
}
```

## for循环与伪代码

**例 2.15**

```
#include<stdio.h>
int main()
{
    float GPA;
    int i;
    for(i=0 ; i<3 ; i++)    自增运算符
    {
        scanf(" %f ", &GPA);
        if( GPA > 4.3 )
            printf( "你作弊啦！ " );
        else
            printf( "还要加油哦~" );
    }
    return 0;
}
```

## for循环与伪代码



## for循环与伪代码

### 伪代码（Pseudocode）

- 用传统流程图、N-S图表示算法，直观易懂，但绘制比较麻烦，在设计一个算法时，可能要反复修改，而修改流程图是比较麻烦的，因此，流程图适合表示算法，但在设计算法过程中使用不是很理想。为了设计算法方便，常使用伪代码工具。
- 伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法。伪代码不用图形符号，书写方便，格式紧凑，便于向计算机语言算法过渡。

```
1. For i=0 to i<3
2. 从键盘输入 GPA 值
3. If GPA>=4.3 输出"你作弊啦！"
4. Else 输出"还要加油哦~"
5. End If
6. i=i+1
7. End For
```

## 数组

**数组的作用：**

批量处理数据

**数组的定义：**

数组的名称

数组的大小（长度/数组元素的个数）

数组的基类型（元素的类型）

[ ] :数组运算符
单目运算符
优先级(1)
左结合
不能用( )

定义方式：　**数据类型　数组名[ 整形常量表达式 ];**

int a[ 6 ] = {1,2,3,4,5,6} ;

数组元素：a[0] a[1] a[2] a[3] a[4] a[5]

## 数组

**例 2.16**

```
#include<stdio.h>
int main()
{
    float data[8]={ 9.812,9.806,9.901,9.788,9.853,9.790,9.819,9.787 };
    float sum=0;
    float ave;
    int i;
    for(i=0;i<8;i++)
    {
        sum=sum+data[i];
    }
    ave=sum/8;
    printf("平均值是%f",ave);
    return 0;
}
```

## 数组

**例 2.16**

```
#include<stdio.h>
int main()
{
    float data[8]={ 9.812,9.806,9.901,9.788,9.853,9.790,9.819,9.787 };
    float sum=0;
    float ave;
    int i;
    for(i=0;i<8;i++)
    {
            sum=sum+data[i];
    }
    ave=sum/8;
    printf("平均值是%f",ave);
    return 0;
}
```

**数组越界**

## 数组

**例 2.17**

```
#include<stdio.h>
int main()
{
    float data[4];
    float sum=0,ave;
    int i;
    for(i=0;i<4;i++)
    {
            scanf("%f", &data[i]);
            sum=sum+data[i];
    }
    ave=sum/4;
    printf("平均值是%.2f",ave);   //%.2f 表示输出时只保留小数点后 2 位
    return 0;
}
```

## C程序规范

- 必须包含声明函数的头文件；
- main函数名前面必须有类型int，结尾必须有return语句；
- 变量名应尽可能反映变量中拟存储数据的信息；
- 建议使用空格或TAB进行缩进；
- 程序都应该有适当的注释；
- 分支语句和循环体哪怕只有一句，也要用大括号括起来；
- 函数内大括号的位置应使得代码更紧凑，阅读节奏感更连续。