

第3章 顺序程序设计 (3)



S.IST@XMU

复习回顾

➤ 上次课的内容：

◆ 数据类型

◆ 运算符与表达式

● 算术运算符

● ++与--

● 逗号运算符

● 赋值中的类型转换

◆ 不要小看基础知识，关键时刻很有用.....

—— 午夜程序员的故事 ——

昨天晚上下班回家，一民警迎面巡逻而来。突然对我大喊：站住！然后发生以下对话


民警喝问：int类型占几个字节？！

我（不假思索地）：4个.....

民警挥挥手：你可以走了

我（感到很诧异）：为什么问这样的问题？

民警：深夜还在街上走，不是程序员就是小偷！



程序员是公司的核心员工，鉴于最近需求较多，且有需要加班的趋势。现决定10月24日放假半天，放假时间段为：24:00-6:00。



“逆天” 的强制类型转换运算符

➤ 可以用强制类型转换运算符将一个表达式转换成所需类型

◆ 一般形式：(**类型名**) (表达式)

● 例如：(double) a; (float) (5%3);

y = (int) x%3; 5 / (float) 3;

◆ **注意**：强制类型转换得到的是一个临时值，存放在临时存储单元，运算对象本身的类型和值并没有变化

● 例如：a = (int) x; x的整数部分（即临时值）赋值给a，x本身不变。

强制类型转换运算符的优先级

➤ 因为C语言将强制类型转换运算符视为**一元运算符**，其优先级要高于二元运算符

◆ 例如：

- 编译器会将 `(int) f+d` 解释为 `((int) f)+d`
- 若 `f` 为 `float` 型，`d` 为 `double` 型，`(int) f+d` 的结果类型将是 `double` 型
- 若期望最终得到 `int` 型的计算结果，表达式应改为 `(int) (f+d)`

强制类型转换的作用举例

➤ 求精确平均值

◆ 比如希望求a,b,c三个**整型变量**精确平均值，表达式应写为 `(double) (a+b+c) / 3`，或者 `(a+b+c) / (double) 3`

➤ 防止溢出

◆ 比如 `double d=111111*111111`，等式右边计算将溢出，d得到诡异的结果；若改为 `(double) 111111*111111` 则不会溢出

显式类型转换

➤ **显式类型转换**是指在程序中借助强制类型转换运算符，人为地强制进行类型转换

◆ **注意**：从表示范围大的类型强制转换到表示范围小的类型，有时会丢失精度。

● 例如假设 `float a=2.5, b=2;`

★ 则表达式 `(int) a*b` 的值为4，

★ 而 `(int) (a*b)` 的值为5

★ `(float) (int) a` 的值是2.0

尝试分辨C语言中的语句

➤ `a=100`

◆不是语句而是表达式，像`a=100`；才是语句。

➤ `100`； 和 `123+456`；

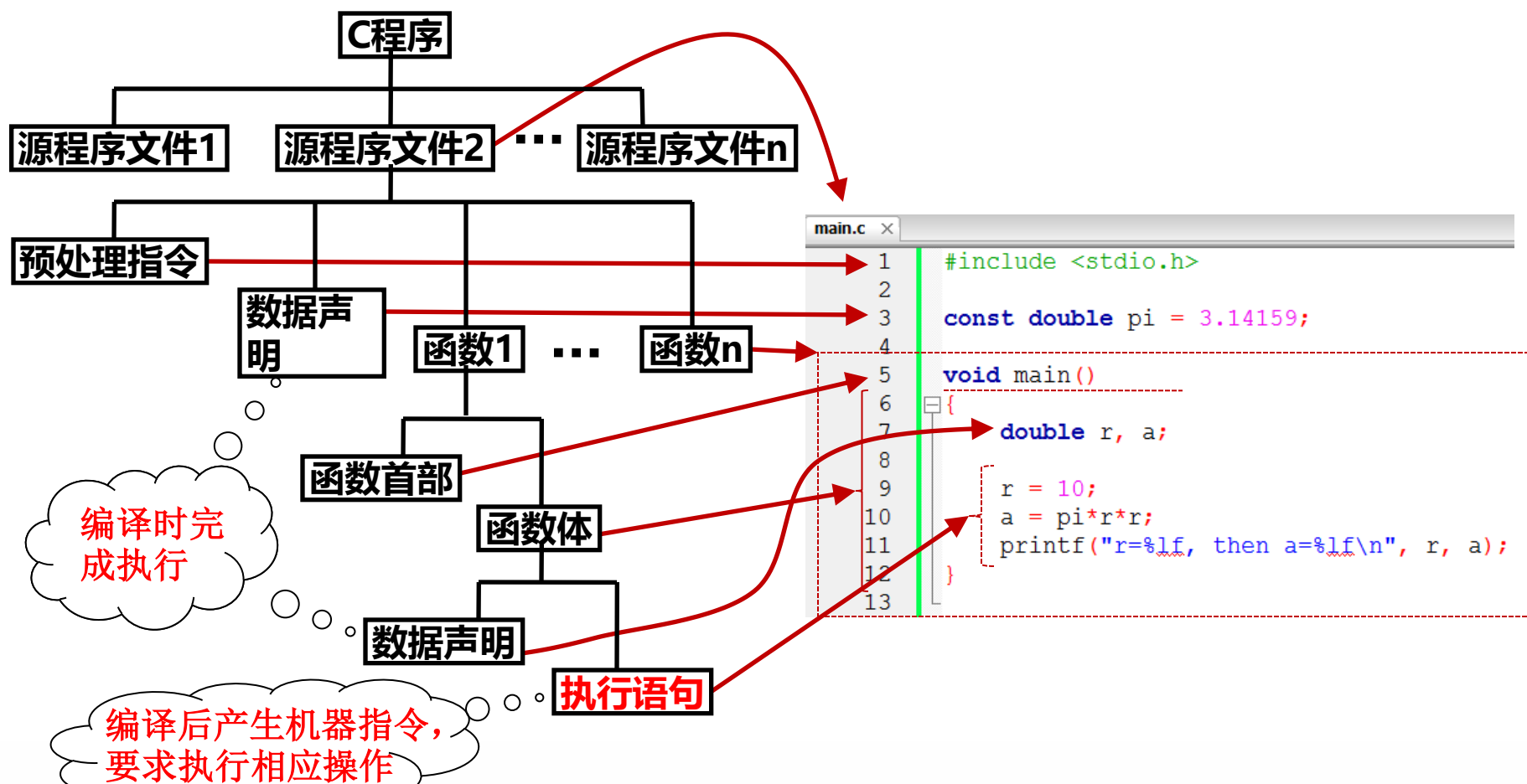
◆都是语句，但都是没有用的语句，因为没有改变任何值，像`a++`； 和 `y=sqrt(x)`； 才是有用的语句。

➤ `y=100-(x++)`；

◆`x++`虽然是完整的指令，但不是一条语句。

◆一条语句是一条完整的指令，但一条完整的指令未必是一条语句。

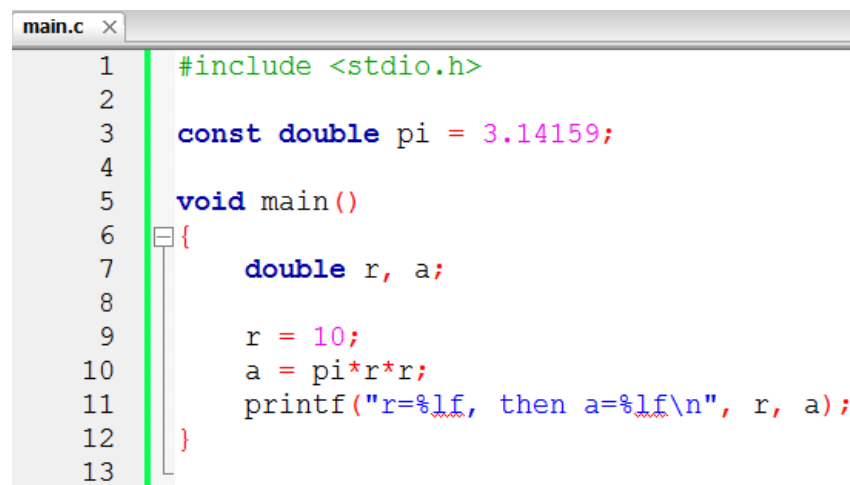
C语句的位置和作用



什么是C语言语句

➤ 源程序包括数据描述部分和数据操作部分

- ◆ **数据描述**主要用来定义数据（用数据类型加变量名表示）和数据初始值
- ◆ **数据操作**对已提供的数据进行加工，这部分代码是由语句来实现的



```
main.c x
1  #include <stdio.h>
2
3  const double pi = 3.14159;
4
5  void main()
6  {
7      double r, a;
8
9      r = 10;
10     a = pi*r*r;
11     printf("r=%lf, then a=%lf\n", r, a);
12 }
13
```

➤ 出现在数据操作部分的带有分号的完整指令才能叫做语句。

C语句的分类

1. **函数调用语句**，如 `printf("Hello");`;
2. **表达式语句**，如 `a=3;`；注意与表达式 `a=3` 的区别
3. **空语句**，即；
4. **复合语句**，如

```
1.  int b;  
2.  {  
3.      int a;;  
4.      a = 1;;  
5.      printf("%d\n",b=++a);  
6.  }
```

5. **控制语句**：条件语句、for循环语句、while循环语句、do...while循环语句、结束本次循环语句、break语句、多分支选择语句、从函数返回语句、转向语句

用一个例子回顾已学到的东西

➤ 给出三角形的三边长，求三角形面积

➤ 解题思路：

◆ 假设给定的三个边符合构成三角形的条件

◆ 关键是找到求三角形面积的公式

◆ 公式为：

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

其中 $s = (a+b+c) / 2$

```
1. #include <stdio.h>
2. #include <math.h>
3. int main ( )
4. {
5.     double a,b,c,s,area;
6.     a=3.67;
7.     b=5.43;
8.     c=6.21;
9.     s=(a+b+c)/2;
10.    area=sqrt(s*(s-a)*(s-b)*(s-c));
11.    printf("a=%f\tb=%f\t%f\n",a,b,c);
12.    printf("area=%f\n",area);
13.    return 0;
14. }
```

基本字符集

特定字

标识符

分隔符

关键字

数据类型

变量声明

常量

变量

执行语句

算术表达式

运算符

输入和输出的概念

- 几乎每个程序都是**数据 “输入-处理-输出”** 的指令集合，所以我们必须掌握输入输出的方法
- 程序的输入输出是**以计算机为主体**
 - ◆ 本课程最常见的方式：键盘输入，显示器输出
- **C语言标准本身不提供输入输出语句**
 - ◆ 所以需要`#include <stdio.h>`
- 程序的输入输出方式可以非常丰富多彩，但是目前我们只使用**`printf`**和**`scanf`**之类的库函数

令人头痛的printf和scanf

➤ **printf一般格式**：printf(“格式控制”，**输出表列**)

◆ 如 `printf("i=%d,c=%c\n",i,c);`

➤ **scanf的一般格式**：scanf(“格式控制”，**地址表列**)

◆ 如 `scanf("%f,%f",&a,&b);`

➤ **使用有风险，书写需谨慎！！！！**

◆ **调用过程**：根据首个字符串参数（即格式控制参数）中的格式说明，从左到右依次输出后续参数的值

◆ **参数个数是不固定的！**

◆ **编译器无法检查参数数据类型的和个数的合法性！！**

printf的基本说明

```
printf("i=%d,c=%c\n", i, c);
```

格式声明

```
printf("i=%d,c=%c\n", i, c);
```

普通字符

```
printf("i=%d,c=%c\n", i, c);
```

输出表列：可以是常量、变量或表达式

格式控制符的组成

➤ 格式控制符的组成

◆ `%` `[- 0 <数字m> . <数字n> l/h]` 类型字符

必须的

有-这个标记就左对齐，没有就右对齐

有0这个标记就填，没有就不填

m是域宽，n是小数位数

l是长，h是短

如 d, f, c, s, o, x, u e, g, i

指定输出类型

长短，修饰数据类型

指定输出域宽及精度

指定左侧空位是否填0

指定是否左对齐输出

格式控制的起始符号

可选

有兴趣的同学自学，
考试不做要求

◆ 如 `double d=3.1415926; printf("PI=%-8.3lf\n", d);`

如何用printf输出整型数据

◆ (1) 十进制形式：以十进制形式输出整型数据：

- %d或%<数字m>d 用于基本整型，如 `int i=-5; printf("%d",i);`
- %ld或%<数字m>ld 用于长整型，如 `long l=99999; printf("%ld",l);`
- %u或%<数字m>u 用于无符号基本整型，如 `unsigned int u=9; printf("%u",u);`
- %lu或%<数字m>lu 用于无符号长整型，如 `long unsigned int n=60000; printf("%lu",n);`

◆ (2) 八进制形式：以八进制形式输出整型数据（无前缀o）：

- %o或%<数字m>o 用于基本整型，如 `int i=9; printf("%o",i);` 输出：11
- %lo或%<数字m>lo 用于长整型，如 `long int x=15; printf("%lo", x);` 输出：17

◆ (3) 十六进制形式：以十六进制形式输出整型数据（无前缀0x）：

- %x或%<数字m>x 用于基本整型，如 `int i=17; printf("%x", i);` 输出：11
- %lx或%<数字m>lx 用于长整型，如 `long int x=257; printf("%lx", x);` 输出：101

◆ 另外，<数字m>表示输出的整型数据所占最小宽度（即列数）。在右（左）对齐模式下，数据的位数小于m，则左（右）端补空格，若大于m，则按实际位数输出。如果在格式控制符中没有用m来说明数据所占的宽度，则以输出数据的实际位数为准

- 如：`printf("%4d,%4d",a,b);` 若 `int a=12,b=12345,`
则输出结果为： □□12,12345（12前面两个空格）

如何用printf输出浮点型数据

◆ 若以小数形式输出浮点型数据，常用格式控制符：`%f` 或 `%<数字m>.<数字n>f`

其中，`<数字m>`表示输出数据所占的总宽度，包括小数点所占的一列，`<数字n>`表示小数点后面的位数。如不指定，默认输出全部整数部分及6位小数。不管`n`的值多大，输出的数字并不是绝对精确的。`float`仅保证6位有效数字，`double`仅保证15位。

【注意】 在C语言中，用于输出单精度浮点型数据和双精度浮点型数据的格式控制符是一样的。`double`型数据输出使用`%lf`和`%f`无区别。

```
1. #include <stdio.h>
2. int main()
3. {
4.     float f = 234.567;
5.     printf("%f\n", f);
6.     printf("%10.2f\n", f);
7.     printf("%-10.2f\n", f);
8.     printf("%.2f\n", f);
9.     printf("%11f\n", f);
10.    return 0;
11. }
```

```
234.567001
□□□□234.57
234.57□□□□
234.57
□234.567001
```

```
// 系统默认输出全部整数和6位小数
// 占10位，小数点后两位，右对齐
// 占10位，小数点后两位，左对齐
// 全部整数部分，小数点后两位
// 占11位，浮点数在内存中存在误差
```

如何用printf输出字符型数据

◆ 其格式控制符为：`%c` 或 `%<数字m>c`

其中，<数字m>表示输出的宽度，在输出字符的左边将要补m-1个空格

例如：`char c='A'; printf("%5c",c);` 输出结果为 "□□□□A"

【注意】 一个整数，只要它的值在0-127范围内，可以用字符形式输出，在输出前，系统会将该整数作为ASCII码转换成相应的字符；反之，一个字符数据也可以用整数形式输出。

```
1.#include <stdio.h>
2.int main()
3.{
4.    char c = 'B';
5.    int i = 66;
6.    printf("%c, %d\n", c, c);
7.    printf("%c, %d\n", i, i);
8.    return 0;
9.}
```

B,	66
B,	66

如何用printf输出字符串数据

◆其控制符可用下列几种方式

(1) %s , 按紧凑格式输出字符串

例如：`printf("%s,%s", "China", "□Beijing");`

输出字符串：`China,□Beijing` (定界符双引号不输出)

- (2) %<数字m>s , 字符串占m列, 长于m则全部输出, 否则右对齐补足空格
- (3) %-<数字m>s , 字符串占m列, 长于m则全部输出, 否则左对齐补足空格
- (4) %<数字m>.<数字n>s , 字符串从左端始截取n个字符, 以%<数字m>s方式输出
- (5) %-<数字m>.<数字n>s , 字符串从左端始截取n个字符, 以%-<数字m>s 方式输出

printf的典型错误举例

```
1.#include <stdio.h>
2.int main()
3.{
4.    int a=20;
5.    double f=1.0;
6.    printf("a=%d,a=%d\n",a);
7.    printf("f=%d\n",f);
8.    printf("f=%d\n", (int) f);
9.    return 0;
10.}
```

a=20 , a=0
f=0
f=1

➤ 格式说明个数和后续参数个数不匹配

◆如以上程序第6行

➤ 格式转换符与实际参数的数据类型不一致

◆如以上程序第7行

scanf的基本说明

```
scanf ("%d, %c", &i, &c);
```

格式声明，同printf

```
scanf ("%d, %c", &i, &c);
```

分隔字符，输入时用作分隔符

```
scanf ("%d, %c", &i, &c);
```

地址表列：可以是变量的地址，或字符串的首地址

scanf的格式控制字符串

➤ **类型说明符**：如`%d`, `%f`等

◆ 含义与`printf`函数中一致

➤ **空白字符**：如`scanf ("%d %d", &a, &b);`

◆ 空白字符会使`scanf`函数在读操作中略去输入中的一个或多个空白符

➤ **非空白字符**：如`scanf ("a=%d", &a);`

◆ 非空白字符会使`scanf`函数在读入时剔除与非空白字符序列相同的字符

scanf读取整数/浮点数的方式

- **首先**，若遇到空白字符（包括空格、制表符和换行符）则跳过直到碰到一个数字字符（小数点也视为数字字符）或正负号
- **若发现一个数字字符**，就保存并读取下一字符
 - ◆ 若接下来仍然是数字则持续保存和读取；
 - ◆ 直到遇到一个非数字字符，放弃此非数字字符并结束该数值输入的读取，计算取到数字的相应数值并放入指定的变量
- **下一个数字读取**从前面被放弃的那个非数字字符后面开始

scanf读取字符/字符串的方式

➤ 若使用%c读取字符

- ◆ 依照%c的个数（假设为n），前n个输入字符都视为有效字符，即使其中有空白字符（包括回车键）也不会跳过，而是将其赋值给变量，只当输入满n个字符后按下回车键输入才会结束；

➤ 若使用%s读取字符串

- ◆ 空白字符以外的所有字符都被视为有效字符，所以将跳过空白字符直到第一个非空白字符，然后保存再次碰到空白字符之前的所有非空白字符。
- ◆ 例如，输入" hi there"，那么保存的第一个字符串是 "hi"

使用scanf时应注意问题之一

- 格式控制后面应当是**变量地址**，若使用变量名则编译检查不到错误，但运行会出错

```
scanf ("%f%f%f", a, b, c);
```

错

```
scanf ("%f%f%f", &a, &b, &c);
```

对

据无责任统计，常年雄霸“初学者常见错误榜”榜首！



使用scanf时应注意问题之二

- 不要画蛇添足！scanf函数**没有精度控制**，
也不需要在末尾加'**\n**'！
 - ◆ 如 `scanf("%6.3f", &f);` 是非法的
 - ◆ 如 `scanf("%f\n", &f);` 末尾的'**\n**'完全没必要，反而会添乱

使用scanf时应注意问题之三

- 如果在“格式控制字符串”中除了格式声明外还有其他字符，则在输入数据时在对应的位置上应输入与这些字符完全相同的字符。所以，除了分隔符尽量不要包含其他字符

比如 `scanf("a=%f,b=%f,c=%f",&a,&b,&c);`

1 3 2✓

错

a=1,b=3,c=2✓

对

a=1 b=3 c=2✓

错

如果 `scanf("%d: %d: %d",&h,&m,&s);`

12: 23: 36✓

对 (两个数据间应有一个冒号和一个以上的空格)

12:23:36✓

错

使用scanf时应注意的问题之四

- 在用“%c”格式输入字符时，所有的字符都作为有效字符输入。

比如 `scanf("%c%c%c", &c1, &c2, &c3)` 试图读取 'a'、'b'、'c' 三个字符；

abc✓ 对 结果 `c1='a', c2='b', c3='c'`

a b c✓ 错 结果 `c1='a', c2=' ', c3='b'`

a✓b✓c✓ 错 结果 `c1='a', c2='\n', c3='b'`

使用scanf应注意的问题之五

- 在输入数据时，如输入空格、回车、Tab键或遇不属于数值的字符，则认为该输入输入结束

比如 `scanf ("%d%c%f", &a, &b, &c);`

如果 1234a1230.26✓

结果 `a=1234, b='a', c=1230.26`

如果 1234a123o.26✓

结果 `a=1234, b='a', c=123`

使用scanf应注意的问题之六

- **若格式控制字符串中的类型说明符与相应的变量类型不一致，虽然编译能通过，但结果将不正确**

比如

```
double f; scanf ("%f", &f); printf ("f=%f\n", f);
```

如果 1.23✓

结果

f=-925596043036449750ooo.oooooo

注意：scanf调用 应改为 `scanf ("%lf", &f)`

putchar函数说明

➤ putchar函数的调用形式

◆ `putchar(c);`

其中，参数c可以是字符型常量、字符型变量或整型变量

➤ putchar函数的功能

◆ 在命令行窗口光标位置处，显示参数c所表示的一个字符。

```
1. #include <stdio.h>
2. int main()
3. {
4.     char a,b,c;
5.     a='O'; b='K'; c='!';
6.     putchar(a); putchar(b); putchar(c);
7.     putchar('\n');
8.     putchar('\x41'); putchar(66);
9.     return 0;
10. }
```

OK!

AB

getchar的使用说明

➤ 字符输入函数getchar

◆调用形式：`getchar()`； //注意：不需要参数

◆功能：接收从键盘输入的一个字符（包括空白字符）

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c; int i;
5.     c=getchar(); printf("c=%-4c", c);
6.     i=getchar(); printf("i=%-3d", i);
7.     printf("c1=%-4c",getchar());
8.     return 0;
9. }
```

abc✓

c=a□□□i=98□c1=c□□□

a✓

c=a□□□i=10□b✓
c1=b

【试验】 如果采用abc✓的输入方式，在第7行之后加一句
`printf("r=%d\n",getchar());`，将会得到什么输出结果？

putchar结合getchar的例子

➤ 从键盘输入一个大写字母在显示屏上显示对应的小写字母

解题思路：

1. 用getchar函数从键盘读入一个大写字母
2. 把它转化为小写字母
3. 用putchar输出该小写字母

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c1, c2;
5.     c1 = getchar();
6.     c2 = c1 + 32;
7.     putchar(c2);
8.     putchar('\n');
9.     return 0;
10. }
```

B ↙

b

【思考】 如果忘了ASCII码表上对应的大小写字母的差值32怎么办？

构思程序结构就像规划行进路线

新闻 网页 贴吧 知道 音乐 图片 视频 地图 百科 文库



前埔医院



厦门大学(海韵校区)



百度一下

下载百度手机地图

搜索

公交

驾车

首页

收藏夹

厦门市 多云 30 ~ 22°C 优 50



百度导航新版调查

点击参与

你吐槽，我送礼

推荐的路线满意吗？给个评价吧！

推荐路线

最短路程

不走高速

约9.7公里/15分钟

发送到手机



起 前埔医院

☐ 显示全部详情



1. 从起点到环岛干道

收起



1) 从起点向正东方向出发，沿文兴东路行驶620米，右转进入环岛干道

2. 环岛干道到终点

收起



1) 沿环岛干道行驶3.3公里，直行进入黄厝隧道

2) 沿黄厝隧道行驶1.1公里，直行进入金山寨隧道

3) 沿金山寨隧道行驶2.1公里，直行进入曾厝垵路

4) 沿曾厝垵路行驶1.6公里，朝曾厝垵西路方向，稍向右转上匝道

5) 沿匝道行驶120米，直行进入曾厝垵北二路



流程控制

- 计算机语言的流程控制就是程序中每个操作执行的先后顺序
- C语言中的基本流程控制（三种）
 - ◆ 顺序：类似于通天大道
 - ◆ 分支：类似于岔路口
 - ◆ 循环：类似于环岛
 - ◆ 跳转：类似于传送门（配合基本流程使用，已少见）
- 基本流程控制的共同特点
 - ◆ 具有一个入口和一个出口



流程的自然语言表示法

➤ 例：求两个数中的最大数

1. 定义两个变量x、y存放两个要比较的数
2. 定义一个变量z存放两个数中的最大数
3. 用户输入x、y的值
4. 判断x的值是否大于y
 - a) 如果满足条件则将x的值赋给z
 - b) 如果不满足条件则将y的值赋给z
5. 输出z的值

➤ 缺点

- ◆ 烦琐：一个很简单的程序就需要很长的一段文字
- ◆ 容易引起歧义，如“小李看到小张在用他的电脑写程序”

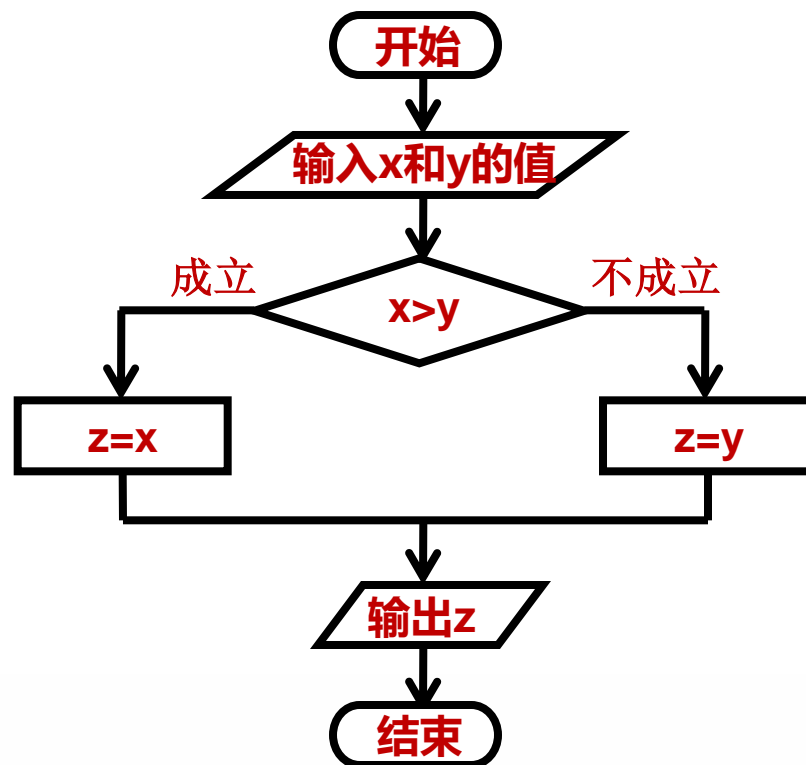
流程图表示法



➤ 例：求两个数中的最大数

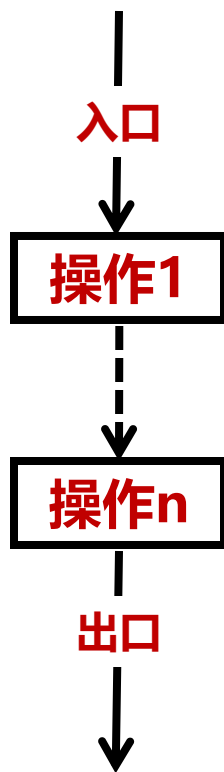
➤ 优点

- ◆ 采用规范的符号，画法简单
- ◆ 结构清晰，逻辑性强
- ◆ 便于描述，容易理解，不容易引起歧义



什么是顺序结构

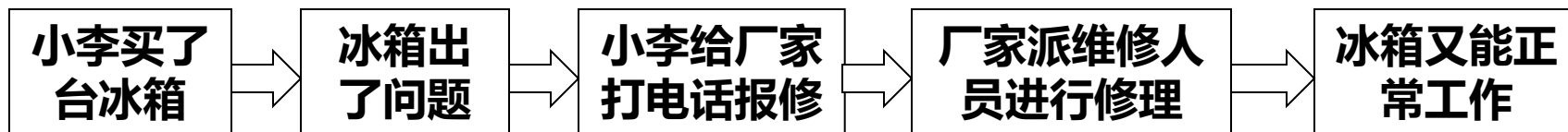
➤ 即程序按照操作出现的先后次序依次执行



那些年，我们一起学过的语文...

➤ 将下列分句按逻辑关系组合成一个完整的句子

- ◆ 冰箱出了问题。
- ◆ 冰箱又能正常工作了。
- ◆ 厂家派维修人员进行修理。
- ◆ 小李给厂家打电话报修。
- ◆ 小李买了台冰箱。



➤ C语言的顺序结构就是把类似于分句的C语句按某种逻辑顺序组合形成一个完整的程序或程序段。

顺序结构程序设计方法

1. 按照题目要求**定义若干变量和常量**；
2. 按照题目要求画出程序的流程图，整理自己的**编程思路**；
3. 按照自己的编程思路**编写C语句**，必要时考虑使用库函数；
4. 根据流程图将数据定义和各条语句**进行组合**，并加上需要包含的头文件等其他内容，**形成完整的程序**。

顺序设计方法演示：温度转换1

➤ 输入华氏温度 f ，输出对应的摄氏温度 c ，
保留三位小数（提示： $c = 5/9 * (f - 32)$ ）

◆ 第一步：定义变量（数据类型是关键）

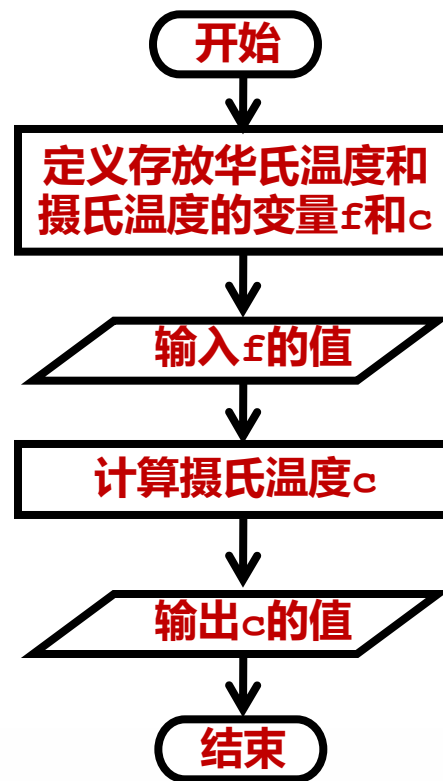
- 由于测量的华氏温度可能有小数部分，所以定义一个 `float` 型变量 f 。
- 而 `float` 型数据参与运算后自动转为 `double` 型，所以需要有一个 `double` 型的变量 c 用来存放转换结果。

顺序设计方法演示：温度转换2

- 输入华氏温度 f ，输出对应的摄氏温度 c ，保留三位小数（提示： $c = 5/9 * (f - 32)$ ）

◆ 第二步：画程序流程图

- 根据流程图知识和题目的要求，画出如右流程图



顺序设计方法演示：温度转换3

➤ 输入华氏温度 f ，输出对应的摄氏温度 c ，保留三位小数（提示： $c=5/9*(f-32)$ ）

◆ 第三步：写出语句

- **输入**：由于 f 是一个实数，所以选择 `scanf ("%f", &f)` 进行输入。
- **计算**：通过计算公式，可以写出求摄氏温度的语句为 `c=5.0/9*(f-32);`。
- **输出**：最后要保留3位小数输出，则输出语句为 `printf ("c=%.3lf", c);`

顺序设计方法演示：温度转换4

➤ 输入华氏温度 f ，输出对应的摄氏温度 c ，保留三位小数（提示： $c=5/9*(f-32)$ ）

◆ 第四步：组合语句，编写程序

```
1. #include <stdio.h>
2. int main()
3. {
4.     float f;
5.     double c;
6.     scanf("%f", &f);
7.     c = (5.0/9) * (f-32);
8.     printf("c=%.3lf\n", c);
9.     return 0;
10. }
```

64✓

c=17.778

【思考】 第7行的算式如果写成 $c=5/9*(f-32)$ 行不行？

顺序结构程序举例：反转1

➤ 输入一个三位数，分离出它的百位、十位和个位，反转后输出。

◆ 样例输入：127

◆ 样例输出：721

【分析】 首先将三位数读入整型变量 n ，然后进行分离：

◆ 个位等于 $n \% 10$

◆ 十位等于 $n / 10 \% 10$

◆ 百位等于 $n / 100$

顺序结构程序举例：反转2

- 输入一个三位数，分离出它的百位、十位和个位，反转后输出。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int n;
5.     scanf("%d", &n);
6.     printf("%d%d%d\n", n%10, n/10%10, n/100);
7.     return 0;
8. }
```

127 ✓
721

【思考】 如果原数的个位数是0，要求反转后最高位不输出0，怎么做？

顺序结构程序举例：反转3

- 输入一个三位数，分离出它的百位、十位和个位，反转后输出。（要求反转后最高位不得输出0）

```
1. #include <stdio.h>
2. int main()
3. {
4.     int n, m;
5.     scanf("%d", &n);
6.     m = (n%10)*100+(n/10%10)*10+(n/100);
7.     printf("%d\n", m);
8.     return 0;
9. }
```

530 ✓
35

【启示】 编程需要严谨的态度。

作业 2017/10/25 (1)

- 完成下列实验，提交纸版实验报告，提供编写的源程序，汇报观察到的现象并试解释其中的原因。

◆ 按照如下数据定义：

- **int** a=100;
- **float** b=123.456;
- **double** c=1234.56789;
- **char** d='s';

使用printf语句，对a, b, c, d四个变量，分别尝试用%d, %x, %o, %c, %f, %u六种方式输出。

例如：
`printf("%d,%x,%o,%c,%f,%u\n", a,a,a,a,a,a);`
`printf("%d,%x,%o,%c,%f,%u\n", b,b,b,b,b,b);`
.....

作业 2017/10/25 (2)

- **完成下列实验，提交纸版实验报告，汇报观察到的现象并试解释原因**
如果用语句`scanf("%d%d",&a,&b)`来输入两个数，那么这两个数应该以怎样的格式输入呢？
- **实验B1：在同一行中输入3和5，并以空格分隔，是否得到预期的结果？**
 - **实验B2：在不同的两行中输入3和5，是否得到了预期的结果？**
 - **实验B3：在实验B1和B2中，在3和5的前面和后面加入大量的空格和水平制表符（TAB），甚至插入些空行，是否能得到预期的结果？**
 - **把5换成字符s，重复实验B1 ~ B3**
- **上机练习（不用交）：编译运行本讲义例程**