项目3

顺序结构

项目描述

在使用计算机处理问题时，程序员需要把问题编写成相应的解决方法和步骤，即算法，而程序可以表述为“算法＋数据结构”。

能力目标

(1) 熟悉各类C语句的表现形式及功能。
(2) 掌握printf函数和scanf函数的格式与使用方法。
(3) 掌握程序设计的三种基本结构及执行过程。
(4) 掌握不同分支语句的使用。

职业素养

(1) 培养分析问题、解决问题的能力。
(2) 培养一定的逻辑思维。
(3) 培养认真负责的工作态度。

任务描述

小明工作于上海某公司，在驾车前往苏州出差的路上，突然接到上级领导电话，目的地改为南京。导航显示距离南京目的地还有210km(该车每百千米耗油7.6L)，汽车仪表已显示油量不足，需要加油(92号汽油)。当时油价为7.73元/L。为保证顺利到达目的地，编程输出至少需要加注的油量及付款金额。

学习目标

(1) 各类C语句。
(2) C语言中格式输入输出函数。
(3) 程序设计的三种基本结构。
(4) 顺序结构程序设计。

3.1 C语句概述

语句是程序中具有确切含义的代码，是构成程序的基本单位。程序的功能就是通过一条条语句的执行而得以实现的。和其他高级语言一样，C语言中的语句用来向计算机

系统发出操作指令,每条语句经编译后产生若干条机器指令,一个程序应当包含若干条语句。应当指出的是,C 语句都是用来完成一定操作任务的,严格地讲声明部分只是对变量的定义,不产生机器操作,其内容不应称为语句。

根据可执行语句的表现形式及功能的不同,可以把 C 语言的语句划分为 5 类。

1. 表达式语句

在 C 语言中,可以由表达式加一个分号构成一个语句。最典型的是,由赋值表达式构成的赋值语句,如 a=97 是一个赋值表达式,而"a=97;"是一个赋值语句。其语句格式如下:

```
表达式;
```

例如:"z=x+y;""i++;""a+b;"等都是表达式语句。

由此可见,任何表达式都可以加上分号而成为语句,分号是语句中不可缺少的一部分。由于 C 程序中大多数语句是表达式语句,所以有人把 C 语言称为"表达式语言"。

2. 函数调用语句

由一次函数调用加一个分号构成一个语句,函数调用语句其实也是一种表达式语句,只因为函数在 C 程序中的独特地位,在此便将函数调用语句单独作为一种语句介绍。其语句格式如下:

```
函数名(参数列表);
```

例如:

```
printf("This is a C program !");
scanf(" %d",&x);
```

C 程序的主体是函数,而函数的使用除了在表达式中出现外,主要是通过函数调用语句使用,所以函数调用语句会在 C 程序中频繁出现,其内容将在后续项目中详细讲述。

3. 流程控制语句

流程控制语句主要是对程序的执行过程起控制作用。一般程序不可能都按顺序执行,有时需要根据不同条件执行不同的语句,这就需要借助流程控制语句实现。C 语言有以下 9 种流程控制语句。

条件判断语句:if 语句、switch 语句。

循环语句:for 语句、do-while 语句、while 语句。

转向语句:goto 语句、break 语句、continue 语句、return 语句。

4. 空语句

空语句即只有一个分号的语句。它什么也不做,只是形式上的语句,主要用作被转折点,或循环结构语句中(表示循环体什么都不做)。其语句格式如下:

```
;
```

例如：

```
while(getchar()! = '\n')
;
```

本程序的功能：只要键盘上输入的不是 Enter，则需重新输入，这里的循环体为空语句。

5. 复合语句

C 语言中允许把一条或多条语句用一对{}括起来，称为复合语句。

例如：

```
{int a,b,c;
 sum = a + b + b; aver = sum/3.0;
 printf("sum = %d,aver = %f",sum,aver);
}
```

复合语句在语法上是一个整体，相当于一个语句。凡是能使用简单语句的地方，都可以使用复合语句。一条复合语句中又可以包含另一条或多条复合语句，在复合语句中还可以定义变量。

注：复合语句{}中各语句都必须以分号结尾，最后一个语句的分号也不能省略，而且在{}之外不能再加分号。

例 3-1　C 语句应用举例。

```
#include "stdio.h"
main()
{
    int x,y;
    x = 65;
    y = x + 32                          //注意此处没有分号，不是语句，编译时会报错!
    printf("x = %c,y = %c",x,y);
}
```

编译时报错，如图 3-1 所示。

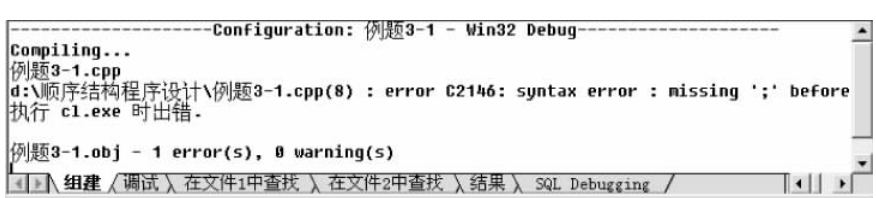

图 3-1　编译时报错

对于本例，y＝x＋32 后面缺少一个分号，没有构成语句，在 C 语言编译时无法通过，会报出如图 3-1 所示的错误。将表达式修改为“y＝x＋32；”，即得到所要的结果。

3.2 赋值语句

赋值语句是由赋值表达式加上一个分号构成。C 语言的赋值语句具有其他高级语言赋值语句的一切特点和功能，但也应注意到它们的不同。

(1) C 语言中的赋值号“＝”是一个运算符，其他大多数语言中赋值号不是运算符。特别说明，C 语言中是用“＝＝”表示等于号。

(2) 关于赋值表达式与赋值语句的概念，其他多数高级语言没有“赋值表达式”这一概念。作为赋值表达式可以包括在其他表达式中，例如“if((a＝b)＞0) t＝a；”。

按语法规定 if 后面的()内是一个条件，其作用是：先进行赋值运算(将 b 的值赋给 a)，然后判断 a 是否大于 0，若大于 0，则执行 t＝a。

在 if 语句中的 a＝b 不是赋值语句而是赋值表达式，如果写成“if((a＝b；)＞0) t＝a；”就错了。在 if 条件中不能包含赋值语句。

由此可以看到，C 把赋值语句和赋值表达式区别开来，增加了表达式的种类，使表达式的应用几乎“无孔不入”，能实现其他语句中难以实现的功能。

3.3 输入与输出语句

输入/输出是对计算机而言，将外部信息送入计算机的过程称为“输入”，将计算机信息送出的过程称为“输出”。数据的输入/输出操作是计算机中重要操作之一，只有通过输入/输出操作，人们才能实现与计算机交换数据。

C 语言本身不提供输入/输出语句，输入/输出操作是通过函数调用实现的。在 C 标准函数库中提供了一些输入/输出函数，如 printf 函数、scanf 函数、putchar 函数、gerchar 函数、puts 函数、gets 函数等。

注：*读者在使用上述函数时，千万不要误认为它们是 C 语言提供的“输入/输出语句”。*

在使用 C 语言库函数时，要用编译预处理命令“＃include”将有关的“头文件”包含到源程序文件中(在“头文件”中包含了被调用函数的有关信息)。在使用标准输入/输出库函数时，源程序文件中的编译预处理命令如下：

```
#include "stdio.h"
```

或

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

其中，stdio.h 是 standard input & output 的缩写，它包括了与标准 I/O 库有关的变量定义和宏定义。本项目将详细介绍格式输入/输出 printf 函数和 scanf 函数。

3.3.1　格式输出函数 printf 函数

printf 函数是 C 语言提供的标准输出库函数，其作用是按照指定的格式向终端输出若干个数据。

1. printf 函数的一般调用格式

printf 函数调用语句的格式如下：

```
printf("格式控制字符串",输出项列表);
```

例如：

```
printf("sum = %d,aver = %f",sum,aver);
```

(1) 格式控制字符串是用双引号括起来的字符串，主要用于说明输出项列表中各输出项的输出格式。

(2) 输出项列表是需要输出的一些数据，输出项可以是合法的常量、变量或表达式，可以是一个或若干个，各项间以逗号隔开。

下面的 printf 函数都是合法的。

```
printf("I am a student.\n");              /*无输出项列表*/
printf("%d",3+2);                         /*输出项列表为表达式*/
printf("a = %f,b = %5d\n", a, a+3);       /*多个数据输出*/
```

2. 格式控制字符串

格式控制字符串中通常包括三部分内容：格式指示符、普通字符、转义字符。

(1) 格式指示符。它是由%和格式字符组成，如%d、%c 等，其作用是将输出的数据转换为指定的格式输出，格式指示符必须由%开始。在 C 语言中的格式指示的一般形式如下：

```
%[宽度][.精度]格式字符
```

对于不同类型的数据要用不同的格式字符，常用的格式字符详见表 3-1。

表 3-1　printf 函数常用格式字符及功能

格式字符	功　能
d	以带符号的十进制形式输出整数(正数不输出符号)
f	以小数形式输出单、双精度实数
c	输出单个字符
s	输出字符串
u	以无符号的十进制形式输出整数
o	以八进制形式输出整数(不输出前缀 0)
x 或 X	以十六进制形式输出无符号整数
e 或 E	以指数形式输出单、双精度实数
g 或 G	选用 f 或 e 格式中输出宽度较小的格式，且不输出无意的 0

① d 格式符,用来输出十进制整数。有以下几种用法。

- %d,按整型数据的实际长度输出。
- %md,m 为指定的输出字段的宽度。如果数据的位数小于 m,则左端补以空格;若大于 m,则按实际位数输出。例如:

```
int x = 345; printf(" % 4d", x);
```

输出结果是□345(□表示空格)。

```
int y = 1234; printf(" % 3d", a);
```

输出结果是 1234。

- %ld,当变量定义为长整型时,输出长整型数据。例如:

```
long z = 123789; printf(" % ld",z);
```

此时如果还使用%d 输出,就会发生错误,因为整型数据的范围为−32768～+32767。

② f 格式符,以小数形式输出单、双精度实数。有以下几种用法。

- %f,不指定字段宽度,系统自动指定,使整数部分全部如数输出,并输出 6 位小数。例如:

```
float x = 123.1485; printf(" % f", x);
```

输出结果是 123.148499。

注:上例中输出的数值并非全部都是有效数字,单精度实数的有效位数为 7 位。

- %m.nf,m,为输出数据所占宽度,其中保留 n 位小数。如果数值长度小于 m,则左端补以空格;若数值长度大于 m,则按实际位数输出;小数位大于 n,则四舍五入,不足n 位右边补零。
- %-m.nf,它与%m.nf,m 基本相同,只是使输出的数值向左端靠齐,右端补空格。

例 3-2 实数输出举例。

```
#include "stdio.h"
main()
{
    float y;
    y = 123.1485;              //变量赋初值
    printf(" % f, % 8.2f, % .2f, % -8.2f",y,y,y,y);
}
```

程序输出结果如下:

123.148499,□□123.15,123.15,123.15□□ (□表示空格),如图 3-2 所示。

图 3-2　程序运行结果

③ c 格式符，用来输出一个字符。例如：

```
char z = 'A'; printf(" % c", z);
```

输出结果为大写字母 A。

例 3-3　格式输出举例。

```
# include "stdio.h"
main( )
{ char c = 'x';
  printf("c:dec = % d,oct = % o,hex = % x,ASCII = % c\n",c,c,c,c);
}
```

程序运行结果如图 3-3 所示。

图 3-3　程序运行结果

(2) 普通字符。普通字符就是需要原样输出的字符，照原样输出。例如：

```
printf("sum = % d, aver = % f",sum,aver);
```

上例 printf 函数中双引号内的“sum＝”“aver＝”、逗号、空格都是普通字符，输出时按原样输出。

(3) 转义字符。转义字符是 C 语言中一种特殊的字符常量，即是一个以“\”开头的字符序列。其意思是将反斜杠后面的字符原来的含义进行转换，变成某种另外特殊约定的含义。例如：

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

其中，“\n”中的 n 不代表字母 n，而作为换行符。

3. 使用 printf 函数时的注意事项

(1) 格式字符与输出项的个数要相同、类型要相匹配,否则,可能会输出不正确。

(2) 除了 X、E、G 外,其他格式字符必须用小写字母。

(3) 若无输出项列表,且格式控制字符串无格式字符,则以普通字符输出。

(4) 使用 printf 函数时,需要在源程序中加入 #include "stdio.h"命令。

(5) 如果想输出字符%,则应该在"格式控制字符串"中用连续两个%表示。

例如:

```
printf("%f%%",2.0/3);
```

输出:

```
0.666667%
```

3.3.2 格式输入函数 scanf 函数

scanf 函数是 C 语言提供的标准输入库函数,其作用是按照格式控制字符串指定的格式要求,从终端键盘输入数据,并送到输入项地址列表指定的内存空间。

1. scanf 函数的一般调用格式

```
scanf("格式控制字符串",输入项地址列表);
```

例如:

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

(1) 格式控制字符串:其含义与 printf 函数基本相同,除必需的标点符号外,最好不出现非格式字符串。

(2) 输入项地址列表:类似于 printf 函数的输出项列表,只是在各变量前面加上地址运算符 &,给出了要赋值的各变量的地址。如上例中的"&a"和"&b",表示变量 a、b 的地址。

2. 格式控制字符串

和 printf 函数中格式控制字符相同,以"%"开始,以一个格式字符结束,中间可以插入附件的字符。scanf 函数常用格式字符及功能详见表 3-2。

表 3-2 scanf 函数常用格式字符及功能

格式字符	功　能
d	输入十进制整数
c	输入单个字符
f	输入实型数(小数形式或指数形式)

续表

格式字符	功　　能
o	输入八进制整数
x 或 X	输入十六进制整数
u	输入无符号十进制整数
s	输入字符串
e、E、g、G	与 f 作用相同，e、f、g 可相互替换（大小写作用相同）

例 3-4　用 scanf 函数输入数据。

```
#include "stdio.h"
main()
{ int x,y,sum;
   scanf("%d,%d",&x,&y);            //调用 scanf 函数输入数据
   sum=x+y;
   printf("sum=%d",sum);
}
```

运行按以下情况输入数据。

```
64,35↙(↙代表 Enter)
```

程序运行结果如图 3-4 所示。

图 3-4　程序运行结果

3. 使用 scanf 函数时的注意事项

（1）scanf 函数的输入项地址列表中变量前的地址运算符“&”不可缺少，否则程序运行出错。如“scanf("%d,%d",x,y);”不对，应将“x,y”改为“&x,&y”。

（2）格式字符应与输入项的类型匹配，否则可能会输入不正确。

（3）格式控制字符串中的普通字符必须按原样输入。

例如：

```
scanf("%d,%d",x,y);
```

输入时应按此形式：

```
64,35↙(↙代表 Enter)
```

注：上例中输入两个数据间的逗号必不可少，否则程序输入错误。

(4) 使用 scanf 函数时，可指定输入数据的宽度，但不能规定输入数据的精度。例如，“scanf("%3d,%3d",&x,&y);”是正确的，而“scanf("%10.2f", &z);”是不合法的。

(5) 格式字符应与输入项的个数相同。若格式字符的个数少，则多余输入项未得到(新的)数据；若格式字符的个数多，则多余的格式字符不起作用。

(6) scanf 函数指定输入/输出宽度时，系统可自动按格式截取输入数据。例如：

```
scanf("%3d%3d",&x,&y);
```

输入时若按此形式：

```
123456↙(↙代表 Enter)
```

系统自动将 123 赋值给 x，456 赋值给 y。此方法也可用于字符型数据。

(7) 使用 scanf 函数时，需要在源程序中加入#include "stdio.h"命令。

3.3.3 putchar 函数(单字符输出函数)

1. 函数功能

putchar 函数的功能是将指定表达式的值所对应的字符输出到标准输出终端。表达式可以是字符型或整型，每次只能向终端输出一个字符。

2. 函数的一般格式

```
putchar(ch); /*ch: 需要输出的字符,可以是字符变量,也可以是字符常量(包括转义字符)*/
```

例 3-5 putchar 函数的应用。

```
#include "stdio.h"
main()
{char ch='a';
   putchar(ch);
   putchar(' ');                //输出空格
   putchar('b');
   putchar('\n');
}
```

程序运行结果如图 3-5 所示。