程序的语句与流程控制

任何复杂的算法都可以用顺序,分支和循环三种结构的语言来实现.一个程序如果仅包含顺序,分支和循环三种结构的语言,则称该程序为结构化程序.

语句分类

C程序总体上可分为**声明语句**和**可执行语句**两大类.
声明语句用于声明和定义程序中被处理数据的类型和名称,包括变量声明,函数原型声明,常量定义,数据类型定义等.
可执行语句用于完成对数据的处理和对程序流程的控制,分为**表达式语句,复合语句,选择语句(if和switch),循环语句(while,do-while和for),转移语句(break,continue,goto和return)和标点语句**.其中表达式语句和符合语句属于顺序结构;选择语句属于分支结构;循环语句属于循环结构;标号语句是在以上任何一种语言前加上一个用作标号的标志符和冒号所形成的语言.

表达式语句

表达式语句的语法形式为:
表达式;
赋值,输入,输出都是通过表达式语句实现的.习惯上,赋值表达式语句称为赋值语句;标准输入函数调用表达式语句称为输入语句;标准输出函数调用表达式语句称为输出语句.
仅由一个分号构成的语句称为**空语句**,它不执行任何操作.如果某处语法上需要一条语句,但实际功能上不需要执行任何操作时,可以使用空语句.

复合语句

符合语句是用"{}"括起来的的一组语句,语法上等价于单个语句.其语法格式为:
{
声明部分
语句部分
}
一个复合语句称为一个**程序块**,**函数体**在语法是一个复合语句,因此又称为程序块.
复合语句体中可以嵌套复合语句,但函数定义不能嵌套.
复合语句的作用:

- 用于语法上只允许出现单个语句,而处理上需要执行多个语句的地方;
- 用于改变嵌套if...else语句配对规则;
- 需要声明临时使用的局部变量(定义在函数,代码块,函数形参列表中的变量)时也可以使用复合语句.

条件语句

if语句

if语句根据一个条件的真和假有选择地执行或不执行某些语句.

if语句的语法形式

if语句有**单分支结构和双分支结构**两种形式:

1. 单分支结构:if格式

```
if(表达式)
    语句1
```

2. 双分支结构:if-else格式

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

表达式是选择条件,可以是任意表达式;语句1和2语法上要求是单个语句(如果是多条语句必须用{}括起来).

嵌套的if语句

1. if子句或else子句中又包含if语句时形成嵌套的if语句.
2. 嵌套if语句中else的配对规则:
   else与其前面最靠近的还未配对的if配对,即**内层优先配对原则**
3. else-if结构
   else的子句又是if子句时,形成特殊的else-if结构.

```
if(表达式1) 语句1;
else if(表达式2) 语句2;
...
else if(表达式n) 语句n;
else 语句n+1
```

switch语句

C语言提供了能实现多语言分支控制的switch语句.

switch语句形式

```
switch(表达式1){
    case 常量表达式1:语句序列1;
                    [break;]
    case 常量表达式2:语句序列2;
                    [break;]
    ...
    case 常量表达式n:语句序列n;
                    [break;]
    [default:语句序列n+1;]
            [break;]
}
```

1. switch是多分支选择语句的标志,case和default只能在switch语句中使用,用来引导case子句和default子句.break的功能是**终止switch语句的执行**,使程序执行该switch语句的下一条语句.
2. switch后面的表达式是选择条件(选择表达式).表达式的值必须是整型(可以是任何整型,字符型或枚举型),选择表达式通常是一个整型变量,或是包含变量的整型表达式.
3. case后面的常量表达式的值是选择表达式可能的取值,通常为常量或字符常量,类型应该与选择表达式的类型一致(整型).
4. switch语句体由多个case和至多一个default组成.
5. 每个case后面可以由零个或多个语句,称为一个case子句.当case后面有多个语句时**不必加{}**.

switch语句执行时,先计算表达式的值,然后将表达式的值依次和case常量比较,当与某个case常量的值相等时,则执行该子句.如果遇到break转移语句,则跳出break所在那一层switch语句;否则依次继续执行后面的语句序列,直到switch语句体结束.
如果选择表达式的值与各case常量都不相等,在有default的情况下执行default子句;否则不执行switch中的任何语句,此时switch语句等价于一个空语句.

switch语句使用要点

1. 列出的case应该能包括要处理的所以取值情况,如果不能包括应该用default语句处理余下情况,
2. break跳出switch语句之后,继续执行switch语句后一个语句;而return语句则立即结束函数并返回到调用处(如果是主函数则程序结束).
3. 如果case子句后面没有转移语句,则程序会继续执行后面的情况,直到结束.
4. switch语句允许多种情况执行相同的语句,这类case可以写成一行,其间可以用空格或者制表符分隔,但是不能用逗号分隔.

循环语句

while语句

while语句的形式

```
while(表达式)
    语句;
```

- 表达式称为循环控制条件;语句称循环体,必须是单条语句(复合语句).
- while语句先计算表达式的值,再判断是否执行循环体中的语句.进入循环前应该先给循环变量赋初值,循环体中要改变循环变量的值,防止出现死循环.
- 循环体由多条语句构成时,必须用{}括起来.循环体只有一个分号时,while成空循环.

程序设计举例

见C文件夹中的程序

E.G.

`while((c=getchar())!='?') putchar(c);` 不能写成 `while(c=getchar() !='?') putchar();` .

do-while语句

do-while循环是一种**后测试循环语句**,即只有在循环体中的代码执行之后,才会测试出口条件.

do_while语句形式

```
do
    语句;
while(表达式);
```

首先执行do后面的循环语句;然后计算并测试表达式的值,如果表达式的值**非0**,则重复执行循环体,否则结束循环.
do-while是直到型循环,while是当型循环.do-while循环体至少执行一次.
do-while语句可以用下面等价语句代替:

```
语句;
while(表达式)
语句;
```

程序设计举例

见C文件夹中的程序

E.G.

表达式也可以作为运算符的操作数执行, `while((x/=10)!=0);` .

for语句

对循环变量赋初值,测试循环条件及修改循环变量的值的操作都可以放到for语句的控制部分,甚至循环体中的某些操作也可以放到循环的控制部分.

for语句的形式

```
for(e1;e2;e3)
s
```

- e1用来给**循环变量赋初值**,一般为赋值表达式或者逗号表达式;e2是**循环条件**,通常为关系表达式或逻辑表达式;e3用来**修改循环变量的值**,通常为赋值表达式,逗号表达式或自增自减运算表达式.
- 先计算e1;然后计算并测试e2的值,若e2的值非0,则执行循环体语句;再计算e3并返回计算和测试e2.
- e1仅在进入for语句时执行一次,e3在每次执行完循环体后都计算一次.
- e1除了用于给循环变量赋初值外,还可以给与循环有关的其他变量赋初值;e3除了用于修改循环变量的值外,还可以执行需要在循环体中进行的其他运算.
- 当循环体中不包括continue语句时,for语句可以改写为下面while语句:

```
e1;
while(e2)
{
    s
    e3;
}
```

C99允许在for语句的e1中定义变量并进行初始化,定义变量的作用域限定于该循环语句内,包括表达式e2和e3以及整个循环体范围.

1. 三个表达式可以部分或全部缺省,但无论缺省哪个,它们之间的分号不能省;
2. `for(;e2;) s` 等价于一个形如 `while(e2) s` 的while语句;
3. `for(e1;;e2) s` 和 `for(;;) s` 都是循环控制条件恒成立的循环语句.

转移语句

break语句

break语句的形式为: `break;` .
break语句的用途有

- 用于switch语句中,从中途退出switch语句;
- 用于循环语句中,从循环体中直接退出当前循环.
- 对于嵌套的循环语句和switch语句,break语句的执行只会退出包含break的那一层结构.
- **见C文件夹中的程序**

continue语句

continue语句的形式为: `continue;`

- 该语句只出现在循环语句中,用于终止循环体的本次执行(并非退出循环语句);即在循环体的本次执行中,跳过从continue语句之后直到循环体结束的所有语句,控制转移到循环体的末尾;
- 对于 `while(e) s;` 和 `do s while(e);` ,在执行continue语句之后立马执行对循环控制表达式e的计算和测试;对于 `for(e1;e2;e3) s;` ,则马上执行对e3的求值,然后执行对表达式e2的计算和测试.

return语句

return语句有下列两种形式:

1. 不带表达式的return语句: `return;` ;
2. 带表达式的return语句: `return 表达式;` .

- return语句的功能是==**从被调用函数返回到调用函数**==.
- 不带表达式的return语句只能返回控制,不能返回值,因此只能用于==从无返回值的函数中返回.一般用于从无返回值函数体的中途返回.对于不包含return语句的无返回值函数,当执行完函数体中最后一个语句后会自动返回到调用处.
- 带表达式的return语句在返回控制的同时,将表达式的值返回到调用处,函数调用表达式的值就是返回值.因此,有返回值的函数(非void类型)**必须至少包含一个**带表达式的return语句,且带表达式的return语句只能用于有返回值的函数.
- 此外,一个函数可以包含多个return语句,这种情况下的return语句通常作为选择语句的子句出现,最终执行的只有其中一个.

```
/* sign:返回浮点数符号的函数,包含多个带表达式的return语句 */
int sign(double x)
{
if(x<0) return -1;
else return ((!x)?0:1);
}
```

程序设计

见C文件夹中的程序

嵌套循环

三种循环语句可以相互任意嵌套.

枚举

许多问题有庞大的问题状态空间，并且给出了一些约束条件，要求寻找解空间的一个或多个解，这类问题的求解需要搜索技术。常见的搜索技术有枚举，深度优先搜索，广度优先搜索等。

递推

递推思想和编程方法结合！用计算机表述数学语言！

筛法