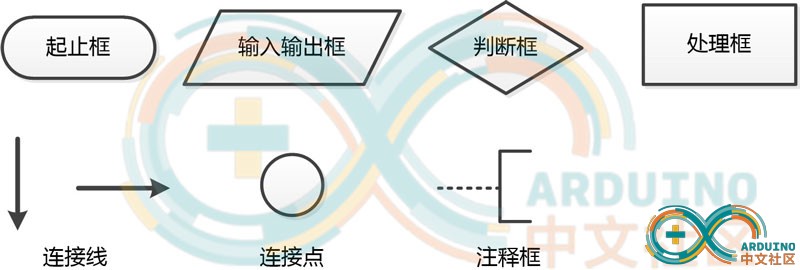
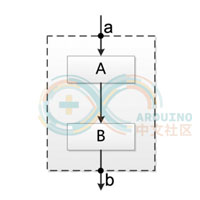
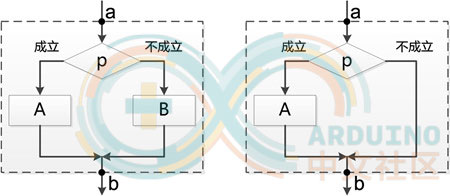
**C\C++语言基础**

**用流程图来表示你的程序**  
       流程图是用一些图框来表示各种操作。用图形表示算法，直观形象，易于理解。特别是对于初学者来，使用流程图能帮你更好的理清思路，从而顺利编写出相应的程序。ANSI规定了一些常用的流程图符号，如图所示：  
  
  
**顺序结构**  
       顺序结构是三种基本结构之一，也是最基本、最简单的程序组织结构。在顺序结构中程序按语句先后顺序依次执行。一个程序或者一个函数，整体上是一个顺序结构，它是由一系列的语句或者控制结构组成，这些语句与结构都按先后顺序运行。  
如图所示，虚线框内是一个顺序结构，其中A、B两个框是顺序执行的。即：在执行完A框中的操作后，接着会执行B框中的操作。  
  
  
  
**选择结构**  
选择结构又称选取结构或分支结构。在编程中，我们经常需要根据当前数据做出判断，决定下一步的操作。例如，Arduino可以通过温度传感器检测出环境温度，在程序中对温度做出判断，如果温度过高，就发出警报信号，这时便会用到选择结构。  
如图所示，虚线框中是一个选择结构。该结构中包含一个判断框。根据判断框中的条件p是否成立，而选择执行A框或者B框。执行完A框或者B框操作后，都会经过b点，脱离该选择结构。  
  
  
**If语句**  
if语句是最常用的选择结构实现方式，当给定表达式为真时，就会运行其后的语句，其有三种结构：  
  
简单分支结构

[C++] *纯文本查看* *复制代码*

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004 | if(表达式)  {  语句;  } |

双分支结构  
双分支结构增加了一个else语句，当给定表达式结果为假时，便运行else后的语句。

[C++] *纯文本查看* *复制代码*

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004  005  006  007  008 | if(表达式)  {  语句1;  }  else  {  语句2;  } |

多分支结构  
使用多个if语句，可以形成多分支结构，用以判断多种不同的情况。

[C++] *纯文本查看* *复制代码*

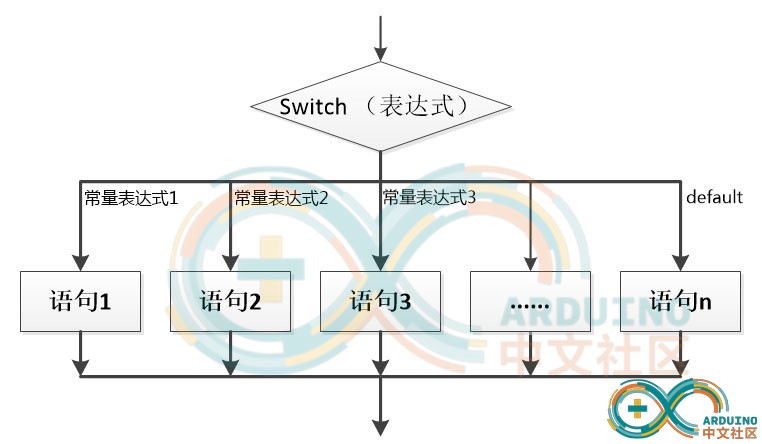
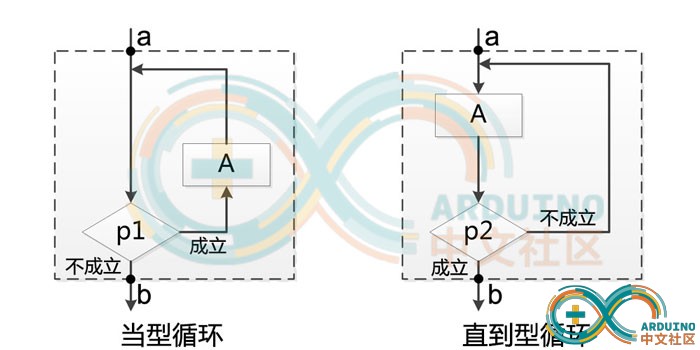
|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017 | if(表达式1)  {  语句1;  }  else  if(表达式2)  {  语句2;  }  else  if(表达式3)  {  语句3;  }  else  if(表达式4)  {  语句4;  }  …… |

**switch…case语句**  
处理比较复杂的问题，可能会存在有很多选择分支的情况，如果还使用if…else的结构编写程序，会使程序显得冗长，且可读性差。  
此时我们可以使用switch，其一般形式为：

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016 | switch(表达式)  {  case 常量表达式1：  语句1           break;  case 常量表达式2：  语句2           break;  case 常量表达式3：  语句3           break;  ……  default ：  语句n           break;  } |

需要注意的是，Switch后的表达式结果只能是整形或字符型。如果要使用其他类型，则必须使用if语句。  
switch结构会将switch语句后的表达式与case后的常量表达式比较，如果符合就运行常量表达式所对应的语句；如果都不相符，则会运行default后的语句，如果不存在default部分，程序将直接退出switch结构。  
在进入case判断，并执行完相应程序后，一般要使用break退出switch结构。如果没有使用break语句，程序则会一直执行到有break的位置退出或运行完该switch结构退出。  
        switch...case结构在流程图中，表示方法如图：  
  
  
  
**循环结构**  
循环结构又称重复结构，即反复执行某一部分的操作。有两类循环结构，当（while）循环和直到（until）循环。  
如图2-5所示。当型循环结构会先判断给定条件，当给定条件p1不成立时，即从b点退出该结构，当p1成立时，执行A框，执行完A框操作后，再次判断条件 p1是否成立，如此反复；直到型循环结构会先执行A框，然后判断给定的条件p2是否成立，成立即从b点退出循环，不成立则返回该结构起始位置a点，重新执行A框，如此反复。  
  
  
  
（1）循环语句  
**while循环**  
while循环是一种当型循环。当满足一定条件后，才会执行循环体中的语句，其一般形式为：

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004 | while(表达式)  {  语句;  } |

在某些Arduino应用中，你可能需要建立一个死循环（无限循环）。当while后的表达式永远为真或者为1时，便是一个死循环。

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004 | while(1)  {  语句;  } |

**do…while循环**  
do…while与while循环不同，是一种直到循环，它会一直循环到给定条件不成立时为止。它会先执行一次do语句后的循环体，再判断是否进行下一次循环。

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004  005 | do  {  语句;  }  while(表达式); |

**for循环**  
for循环比while循环更灵活，且应用广泛，它不仅适用于循环次数确定的情况，也适用于循环次数不确定的情况。while和do…while都可以替换为for循环。  
一般形式为：

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

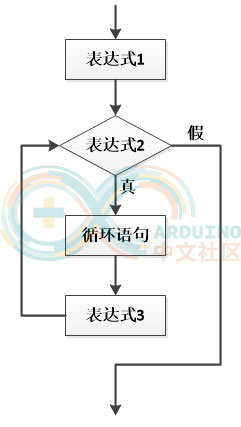
|  |  |
| --- | --- |
| 代码 | |
| 001  002  003  004 | for(表达式1;表达式2;表达式3)  {  语句;  } |

一般情况下，表达式1为for循环初始化语句，表达式2为判断语句，表达式3为增量语句。如

[C++] *纯文本查看* *复制代码*

[?](http://www.arduino.cn/)

|  |  |
| --- | --- |
| 代码 | |
| 001 | for (i=0; i<5; i++) { } |

表示初始值i为0，当i小于5时运行循环体中的语句，每循环完一次，i自加1，因此这个循环会循环5次  
  
  
for循环流程图表示图所示：  
  
图 2 6 for循环流程图  
  
（2）循环控制语句  
在循环结构中，都有一个表达式用于判断是否进入循环。通常情况下，当该表达式结果为false（假）时，会结束循环。有时候，需要提前结束循环，或是已经达到了一定条件，可以跳过本次循环余下的语句，那么可以使用循环控制语句break和continue。  
        breakλ  
break语句只能用于switch多分支选择结构和循环结构中，使用它可以终止当前的选择结构或者循环结构，使程序转到后续的语句运行。break一般会搭配if语句使用。  
一般形式为：  
if(表达式)  
{  
  break;  
}  
  
continue  
continue语句用于跳过本次循环中剩下的语句，并判断是否开始下一次循环。同样，continue一般搭配if语句使用  
一般形式为：  
if(表达式)  
{  
continue;  
}  
  
在编写程序前，你可以先画出流程图，帮助你理清思路。在上一章中，我们所看到的例程Blink，用流程图可表示为图2-7的形式：  
