```
/*==========================================================================
PID 处理例程
这是从网上找来的一个比较典型的 PID 处理程序，在使用单片机作为控制 CPU 时，请稍作
简化，具体的 PID 参数必须由具体对象通过实验确定。由于单片机的处理速度和 RAM 资源
的限制，一般不采用浮点数运算，而将所有参数全部由整数，运算到最后再除以一个 2
的 N 次方数据（相当于移位），作类似定点数运算，可大大提高运算速度，根据控制精
的不同要求，当精度要求很高时，注意保留移位引起的"余数"，做好余数补。这个程序
只是一般常用 PID 算法的基本架构，没有包含输入输出处理部分。
==========================================================================*/
#include<string.h>
#include<stdio.h>
/*==========================================================================
PID Function

The PID(比例，积分，微分）function is used in mainly
control applications.PIDCalc performs one iteration of the PID algorithm.

While the PID function works,main is just a dummy program showing a typical usage.
==========================================================================*/
typedef struct   PID{
double SetPoint;  //设定目标  Desird value

double Proportion;//比例常数  Proportional Const
double Integral; //积分常数   Integral Const
double Derivative; //微分常数  Derivative Const
double LastError;//Error[-1]
double PrevError;//Error[-2]
double SumError;//Sums of Errors
}PID;

/*==========================================================================
PID 计算部分
==========================================================================*/
double PIDCalc(PID *pp,double NextPoint)
{
double dError,Error;
Error=pp->SetPoint-NextPoint;//偏差
pp->SumError+=Error;//积分
dError=pp->LastError-pp->PrevError;//当前微分
 pp->PrevError=pp->LastError;
 pp->LastError=Error;
 return(pp->Proportion * Error//比例项
 +pp->Integral * pp->SumError   //积分项
 +pp->Derivative * dError       //微分项
```

```
 );
}

/*==============================================================================
Initialize PID Structure
==============================================================================*/
void PIDInit(PID *pp)
{
  memset(pp,0,sizeof(PID));
}

/*==============================================================================
Main Program
==============================================================================*/
double sensor(void)//Dummy Sensor Function
{
return 100.0;
}

void actuator(double rDelta)
{}

int main(void)
{
 PID sPID;//PID Control Structure
 double rOut;//PID Response(Output)
 double rIn;//PID Feedback(Input)

 PIDInit(&sPID);//Initialize Structure
 sPID.Proportion=0.5;//Set PID Coefficients
 sPID.Integral=0.5;
 sPID.Derivative=0.0;
 sPID.SetPoint=100.0;//Set PID Setpoint

 for(;;){//Mock Up of PID Processing
  rIn=sensor();//Read Input
  rOut=PIDCalc(&sPID,rIn);//Perform PID Interation
  actuator(rOut);//Effect Needed Changes
 }
}
```