|  |  |  |  |
| --- | --- | --- | --- |
| 测试输入 | 期望输出 | 实际输出 | 备注 |
| const a=10;  var b,c;  begin  read(b);  c:=a+b;  write(c);  end. | (constsym,const)  (ident,a)  (eql,=)  (number,10)  (semicolon,;)  (varsym,var)  (ident,b)  (comma,,)  (ident,c)  (semicolon,;)  (beginsym,begin)  (readsym,read)  (lparen,()  (ident,b)  (rparen,))  (semicolon,;)  (ident,c)  (becomes,:=)  (ident,a)  (plus,+)  (ident,b)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,c)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | (constsym,const)  (ident,a)  (eql,=)  (number,10)  (semicolon,;)  (varsym,var)  (ident,b)  (comma,,)  (ident,c)  (semicolon,;)  (beginsym,begin)  (readsym,read)  (lparen,()  (ident,b)  (rparen,))  (semicolon,;)  (ident,c)  (becomes,:=)  (ident,a)  (plus,+)  (ident,b)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,c)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | **覆盖单词：**  基本字：begin end const read write  运算符：+ :=  界符：( ) , ; . |
| const a=10;  var b,c;  begin  READ(b);  c:=a+b;  write(C);  end. | (constsym,const)  (ident,a)  (eql,=)  (number,10)  (semicolon,;)  (varsym,var)  (ident,b)  (comma,,)  (ident,c)  (semicolon,;)  (beginsym,begin)  (readsym,READ)  (lparen,()  (ident,b)  (rparen,))  (semicolon,;)  (ident,c)  (becomes,:=)  (ident,a)  (plus,+)  (ident,b)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,C)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | (constsym,const)  (ident,a)  (eql,=)  (number,10)  (semicolon,;)  (varsym,var)  (ident,b)  (comma,,)  (ident,c)  (semicolon,;)  (beginsym,begin)  (readsym,READ)  (lparen,()  (ident,b)  (rparen,))  (semicolon,;)  (ident,c)  (becomes,:=)  (ident,a)  (plus,+)  (ident,b)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,C)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | **覆盖单词：**  同上  **说明：**  是否区分大小写 |
| var n, f;  begin  n := 0;  f := 1;  while n # 10 do  begin  n := n + 1;  f := f \* n;  end;  call print;  end. | (varsym,var)  (ident,n)  (comma,,)  (ident,f)  (semicolon,;)  (beginsym,begin)  (ident,n)  (becomes,:=)  (number,0)  (semicolon,;)  (ident,f)  (becomes,:=)  (number,1)  (semicolon,;)  (whilesym,while)  (ident,n)  (neq,#)  (number,10)  (dosym,do)  (beginsym,begin)  (ident,n)  (becomes,:=)  (ident,n)  (plus,+)  (number,1)  (semicolon,;)  (ident,f)  (becomes,:=)  (ident,f)  (times,\*)  (ident,n)  (semicolon,;)  (endsym,end)  (semicolon,;)  (callsym,call)  (ident,print)  (semicolon,;)  (endsym,end)  (period,.) | (varsym,var)  (ident,n)  (comma,,)  (ident,f)  (semicolon,;)  (beginsym,begin)  (ident,n)  (becomes,:=)  (number,0)  (semicolon,;)  (ident,f)  (becomes,:=)  (number,1)  (semicolon,;)  (whilesym,while)  (ident,n)  (neq,#)  (number,10)  (dosym,do)  (beginsym,begin)  (ident,n)  (becomes,:=)  (ident,n)  (plus,+)  (number,1)  (semicolon,;)  (ident,f)  (becomes,:=)  (ident,f)  (times,\*)  (ident,n)  (semicolon,;)  (endsym,end)  (semicolon,;)  (callsym,call)  (ident,print)  (semicolon,;)  (endsym,end)  (period,.) | **覆盖单词：**  基本字 begin end var do while call  运算符：# + \* :=  界符： , ; . |
| var as, fa,x,n,y,z;  begin  as := 0;  n := 45;  fs := 30;  x:= 78;  y :=34;  if as > fa  then  PROCEDURE m;  var sum,first,count;  BEGIN  sum:=first+count\*10  end;  if as <= fa  then  as:=as/fa;  if x >= y  then x:=x-y;  if x = y  then x:=10;  if x < y  then y:=5;  if ODD b THEN z := x + y;  end. | (varsym,var)  (ident,as)  (comma,,)  (ident,fa)  (comma,,)  (ident,x)  (comma,,)  (ident,n)  (comma,,)  (ident,y)  (comma,,)  (ident,z)  (semicolon,;)  (beginsym,begin)  (ident,as)  (becomes,:=)  (number,0)  (semicolon,;)  (ident,n)  (becomes,:=)  (number,45)  (semicolon,;)  (ident,fs)  (becomes,:=)  (number,30)  (semicolon,;)  (ident,x)  (becomes,:=)  (number,78)  (semicolon,;)  (ident,y)  (becomes,:=)  (number,34)  (semicolon,;)  (ifsym,if)  (ident,as)  (gtr,>)  (ident,fa)  (thensym,then)  (ident,PROGRAM)  (ident,m)  (semicolon,;)  (varsym,var)  (ident,sum)  (comma,,)  (ident,first)  (comma,,)  (ident,count)  (semicolon,;)  (beginsym,BEGIN)  (ident,sum)  (becomes,:=)  (ident,first)  (plus,+)  (ident,count)  (times,\*)  (number,10)  (endsym,end)  (semicolon,;)  (ifsym,if)  (ident,as)  (leq,<=)  (ident,fa)  (thensym,then)  (ident,as)  (becomes,:=)  (ident,as)  (slash,/)  (ident,fa)  (semicolon,;)  (ifsym,if)  (ident,x)  (geq,>=)  (ident,y)  (thensym,then)  (ident,x)  (becomes,:=)  (ident,x)  (minus,-)  (ident,y)  (semicolon,;)  (ifsym,if)  (ident,x)  (eql,=)  (ident,y)  (thensym,then)  (ident,x)  (becomes,:=)  (number,10)  (semicolon,;)  (ifsym,if)  (ident,x)  (lss,<)  (ident,y)  (thensym,then)  (ident,y)  (becomes,:=)  (number,5)  (semicolon,;)  (ifsym,if)  (oddsym,ODD)  (ident,b)  (thensym,THEN)  (ident,z)  (becomes,:=)  (ident,x)  (plus,+)  (ident,y)  (semicolon,;)  (endsym,end)  (period,.) | (varsym,var)  (ident,as)  (comma,,)  (ident,fa)  (comma,,)  (ident,x)  (comma,,)  (ident,n)  (comma,,)  (ident,y)  (comma,,)  (ident,z)  (semicolon,;)  (beginsym,begin)  (ident,as)  (becomes,:=)  (number,0)  (semicolon,;)  (ident,n)  (becomes,:=)  (number,45)  (semicolon,;)  (ident,fs)  (becomes,:=)  (number,30)  (semicolon,;)  (ident,x)  (becomes,:=)  (number,78)  (semicolon,;)  (ident,y)  (becomes,:=)  (number,34)  (semicolon,;)  (ifsym,if)  (ident,as)  (gtr,>)  (ident,fa)  (thensym,then)  (ident,PROGRAM)  (ident,m)  (semicolon,;)  (varsym,var)  (ident,sum)  (comma,,)  (ident,first)  (comma,,)  (ident,count)  (semicolon,;)  (beginsym,BEGIN)  (ident,sum)  (becomes,:=)  (ident,first)  (plus,+)  (ident,count)  (times,\*)  (number,10)  (endsym,end)  (semicolon,;)  (ifsym,if)  (ident,as)  (leq,<=)  (ident,fa)  (thensym,then)  (ident,as)  (becomes,:=)  (ident,as)  (slash,/)  (ident,fa)  (semicolon,;)  (ifsym,if)  (ident,x)  (geq,>=)  (ident,y)  (thensym,then)  (ident,x)  (becomes,:=)  (ident,x)  (minus,-)  (ident,y)  (semicolon,;)  (ifsym,if)  (ident,x)  (eql,=)  (ident,y)  (thensym,then)  (ident,x)  (becomes,:=)  (number,10)  (semicolon,;)  (ifsym,if)  (ident,x)  (lss,<)  (ident,y)  (thensym,then)  (ident,y)  (becomes,:=)  (number,5)  (semicolon,;)  (ifsym,if)  (oddsym,ODD)  (ident,b)  (thensym,THEN)  (ident,z)  (becomes,:=)  (ident,x)  (plus,+)  (ident,y)  (semicolon,;)  (endsym,end)  (period,.) | **覆盖单词：**  基本字 begin end var if then procedure odd  运算符：- + \* / := > < >= <= =  界符： , ; . |
| var m, n, r, q;  procedure gcd;  begin  while r#0 do  begin  q := m / n;  r := m - q \* n; m := n;  n := r;  end  end;  begin  read(m);  read(n);  r: 1;  call gcd;  write(m);  end. | (varsym,var)  (ident,m)  (comma,,)  (ident,n)  (comma,,)  (ident,r)  (comma,,)  (ident,q)  (semicolon,;)  (proceduresym,procedure)  (ident,gcd)  (semicolon,;)  (beginsym,begin)  (whilesym,while)  (ident,r)  (neq,#)  (number,0)  (dosym,do)  (rparen,))  (semicolon,;)  (ident,r)  error!(number,1)  (semicolon,;)  (callsym,call)  (ident,gcd)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,m)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | (varsym,var)  (ident,m)  (comma,,)  (ident,n)  (comma,,)  (ident,r)  (comma,,)  (ident,q)  (semicolon,;)  (proceduresym,procedure)  (ident,gcd)  (semicolon,;)  (beginsym,begin)  (whilesym,while)  (ident,r)  (neq,#)  (number,0)  (dosym,do)  (rparen,))  (semicolon,;)  (ident,r)  error!(number,1)  (semicolon,;)  (callsym,call)  (ident,gcd)  (semicolon,;)  (writesym,write)  (lparen,()  (ident,m)  (rparen,))  (semicolon,;)  (endsym,end)  (period,.) | **覆盖单词：**  基本字 begin end var read write procedure  运算符：- + \* / # :=  界符：( ) , ; .  **说明：**  存在运算符:=错误，输出错误提示error |