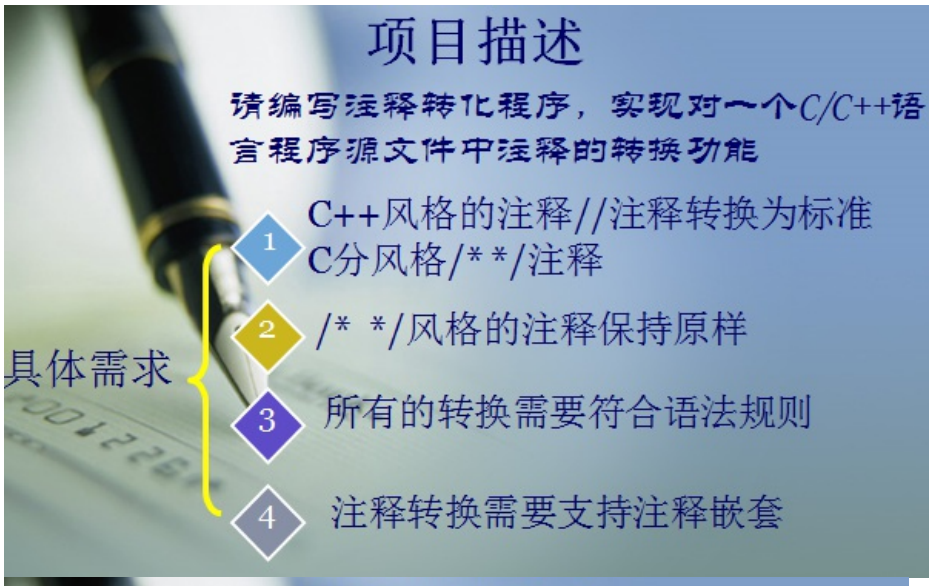
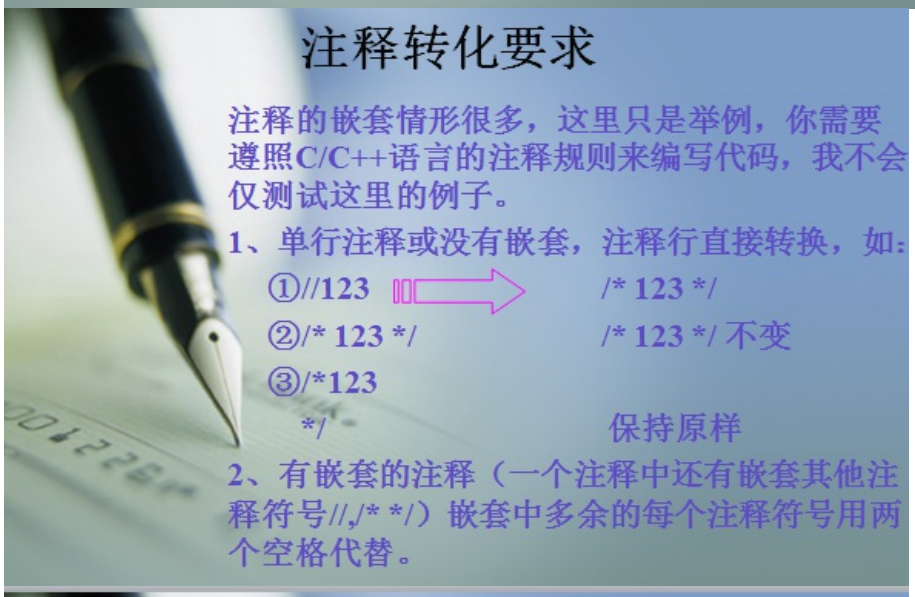
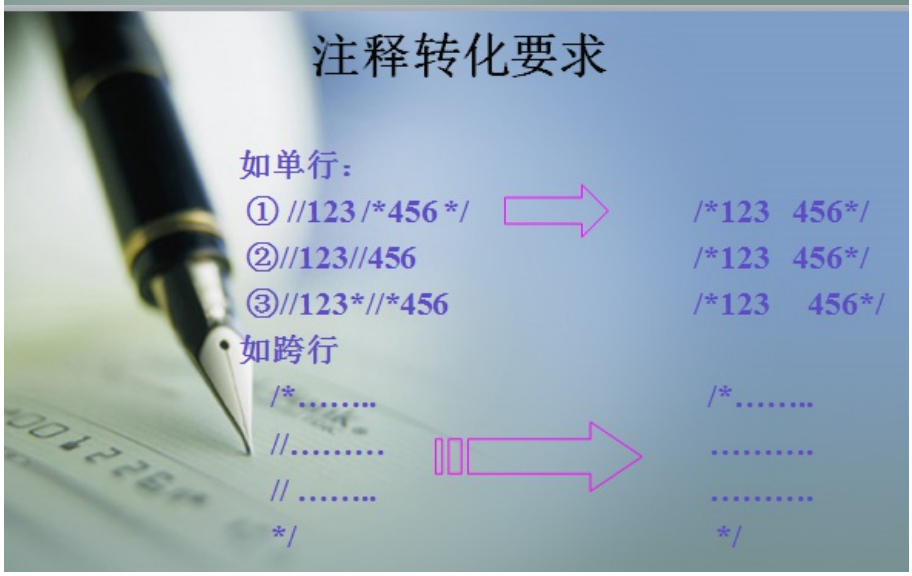
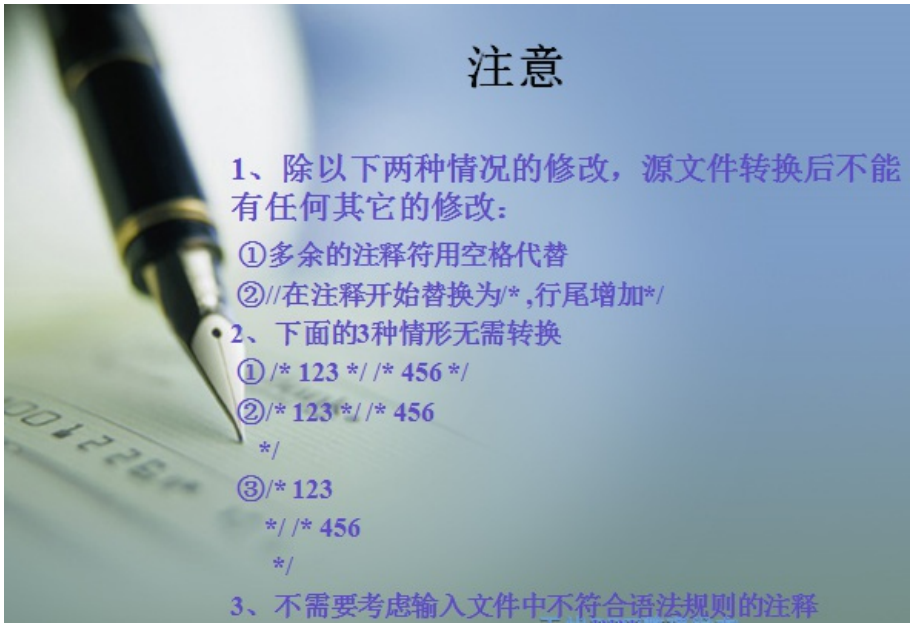
项目一：C++注释转换为C语言注释









进行注释转换在读取源文件的时候，可能遇到的情况有：

1.C 风格注释（包含注释的嵌套）

2.C++ 风格注释（包含注释的嵌套）

3.字符中嵌套注释，无注释 ，结束等几种状态。

利用有限状态机解决注释转换问题

什么是有限状态机？
有限状态机FSM是软件上常用的一种处理方法，它把复杂的控制逻辑分解成有限个稳定状态，在每个状态上进行处理。有限状态机是闭环系统，可以用有限的状态，处理无穷的事务。

```
// 通常我们使用多路分之语句来处理状态机
switch (state)
{
case 1:
     // ...
     break ;
case 2:
     // ...
     break ;
case 3:
     // ...
     break ;
.
.
.
case n :
     break ;
default :
}
```

状态机简写为FSM(Finite State Machine),主要分为2大类:

第一类，若输出只和状态有关而与输入无关，则称为Moore状态机;

第二类，输出不仅和状态有关而且和输入有关系，则称为Mealy状态机。

状态机可归纳为4个要素，即现态、条件、动作、次态。

这样的归纳，主要是出于对状态机的内在因果关系的考虑。"现态"和"条件"是因，"动作"和"次态"是果。

详解如下:

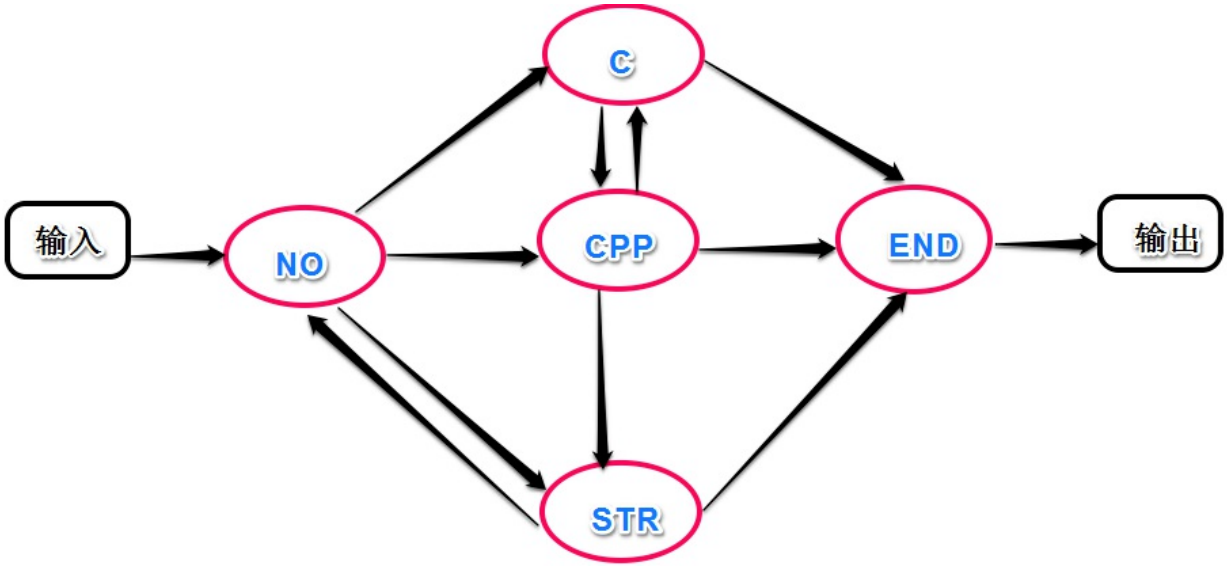
①现态:是指当前所处的状态。

②条件:又称为"事件"，当一个条件被满足，将会触发一个动作，或者执行一次状态的迁移。

③动作:条件满足后执行的动作。动作执行完毕后，可以迁移到新的状态，也可以仍旧保持原状态。动作不是必需的，当条件满足后，也可以不执行任何动作，直接迁移到新状态。

④次态:条件满足后要迁往的新状态。"次态"是相对于"现态"而言的，"次态"一旦被激活，就转变成新的"现态"了。

画出注释转换的状态机状态转换图：



写出测试用例测试用例：

```
//1.C注释
//int i = 0;*/

//*//*int i = 0;

// /**/int i = 0;
2.C++正常注释
/*int i;*/

//3.C++嵌套注释
/* int i = 0;
 *//*
 */
```

```
/* int i = 0;
//*/int j = 0;



/* int i = 0;*//*int j = 0;
 */


/*
 *//*
 */int i =0;


//4多重注释
//////////////////////////////////////////////////////////////////////////////////////////////
//////////////

//////////////////////////////////////////////////////////////////////////////////////////////
//////////////

////// int i;
//5.字符串含有注释
"abcdefghijklmn  //   /*       *\~~~~!!!!!!!!"
//6.字符串
"aaaahfhkahkshfkshkahsfakf"
```

注释转换程序：

```
/**********************************************************
********* File Name: CommentCover.h  *********************
**********************************************************\
#include<iostream>
using namespace std;
#define STACKSIZE 1024
#define UL unsigned long

//extern int CommentConvert(FILE *inputfile, FILE *outputfile);

typedef enum
```

```
{
 NO_COMMENT_STATE,
 C_COMMENT_STATE,
 CPP_COMMENT_STATE,
 STR_STATE,
 END_STATE
}STATE_ENUM;//状态列表

typedef struct
{
 FILE *inputfile;
 FILE *outputfile;
 STATE_ENUM ulstate;
}STATE_MACHINE;  //状态机

//
STATE_MACHINE g_state = { 0 };

////////////////////////////////////////////////
void EventPro(char ch);//事件驱动
void EventProAtNo(char ch);
void EventProAtC(char ch);
void EventProAtCpp(char ch);
void EventProAtStr(char ch);
////////////////////////////////////////////////

int CommentConvert(FILE *inputfile, FILE *outputfile)
{
 if (inputfile == NULL || outputfile == NULL)
 {
  cout << "input argument Invalid!" << endl;
  return -1;
 }

 g_state.inputfile = inputfile;
 g_state.outputfile = outputfile;
 g_state.ulstate = NO_COMMENT_STATE;  //初始状态为无注释状态

 char ch;
```

```
 while (g_state.ulstate != END_STATE)
 {
  ch = fgetc(g_state.inputfile); //
  EventPro(ch);
 }
 return 0;
}

void EventPro(char ch)  //事件驱动模型
{
 switch (g_state.ulstate)   //不同的事件状态使用不同的状态函数
 {
 case NO_COMMENT_STATE:
  EventProAtNo(ch);
  break;
 case C_COMMENT_STATE:
  EventProAtC(ch);
  break;
 case CPP_COMMENT_STATE:
  EventProAtCpp(ch);
  break;
 case STR_STATE:
  EventProAtStr(ch);
  break;
 case END_STATE:
  break;
 }
}

void EventProAtNo(char ch)
{
 char nextch;
 switch (ch)
 {
 case '/':   // // /*
  nextch = fgetc(g_state.inputfile);
  if (nextch == '/') // C++
  {
   fputc('/', g_state.outputfile);
```

```
    fputc('*', g_state.outputfile);  //将CPP的//转化为/*
    g_state.ulstate = CPP_COMMENT_STATE;//转换为CPP状态
   }
   else if (nextch == '*') //C
   {
    fputc(ch, g_state.outputfile);
    fputc(nextch, g_state.outputfile);
    g_state.ulstate = C_COMMENT_STATE;//转换为C状态
   }
   else
   {
   }
   break;
  case EOF:
   g_state.ulstate = END_STATE;
   break;
  case '"':
   fputc('"', g_state.outputfile);
   g_state.ulstate = STR_STATE;

   break;
  default:
   fputc(ch, g_state.outputfile);
   break;
  }
}
void EventProAtC(char ch)
{
 char nextch;
 switch (ch)
 {
 case '*':
  nextch = fgetc(g_state.inputfile);
  if (nextch == '/')
  {
   fputc(ch, g_state.outputfile);
   fputc(nextch, g_state.outputfile);
   g_state.ulstate = NO_COMMENT_STATE;
  }
```

```
  break;
 case '/':
  nextch = fgetc(g_state.inputfile);
  if (nextch == '/')
  {
   fputc(' ', g_state.outputfile);
   fputc(' ', g_state.outputfile);//嵌套注释用两个空格代替
  }
  break;
 default:
  fputc(ch, g_state.outputfile);
  break;
 }
}
void EventProAtCpp(char ch)
{
 //123  /*123
 char nextch;
 switch (ch)
 {
 case '\n': //处理多行
  fputc('*', g_state.outputfile);
  fputc('/', g_state.outputfile);
  fputc('\n', g_state.outputfile);
  g_state.ulstate = NO_COMMENT_STATE;
  break;
 case EOF:
  fputc('*', g_state.outputfile);
  fputc('/', g_state.outputfile);
  g_state.ulstate = END_STATE;
  break;
 case '/':
  nextch = fgetc(g_state.inputfile);
  if (nextch == '/') //  (嵌套//)
  {
   fputc(' ', g_state.outputfile);
   fputc(' ', g_state.outputfile);
  }
  else if (nextch == '*') //  (嵌套 /*)
```

```
  {
   fputc(' ', g_state.outputfile);
   fputc(' ', g_state.outputfile);
  }
  else
  {
   fputc(ch, g_state.outputfile);
  }
  break;

 case '*':
  nextch = fgetc(g_state.inputfile);
  if (nextch == '/') //  嵌套//
  {
   fputc(' ', g_state.outputfile);
   fputc(' ', g_state.outputfile);
  }
  else
  {
   fputc(ch, g_state.outputfile);
  }
  break;
 case'"':
  g_state.ulstate = STR_STATE;
 default:
  fputc(ch, g_state.outputfile);
  break;
 }
}


void EventProAtStr(char ch)
{
 char nextch;
 switch (ch)
 {
 case '\0':
  nextch = fgetc(g_state.inputfile);
  if (nextch == '"')      //读取到 \0 和 " 说明字符串结束
```

```
    g_state.ulstate = NO_COMMENT_STATE; //状态切换
   break;
  case EOF:
   g_state.ulstate = END_STATE;
   break;
  default:
   fputc(ch, g_state.outputfile);
   break;
  }
}
/*************************************************************
****************  main.h ************************************
*************************************************************\

#include"CommentConvert.h"
//extern int CommentConvert(FILE *inputfile, FILE *outputfile);

int main()
{
 FILE *fpIn = NULL;  //inputfile
 FILE *fpOut = NULL; //outputfile

 fpIn = fopen("input.c","r");
 if(NULL == fpIn)
 {
  cout<<"Open input file fail!"<<endl;
  return -1;
 }


 fpOut = fopen("output.c","w");
 if(NULL == fpOut)
 {
  cout<<"Open output file fail!"<<endl;
  return -1;
 }
 CommentConvert(fpIn,fpOut);

 fclose(fpIn);
```

```
 fclose(fpOut);
 return 0;
}


/*************************************************************
*********   output.c  ************************************
*************************************************************\

/*1.C注释*/
/*int i = 0;                */

/*    int i = 0;            */

/*     int i = 0;           */
2.C++正常注释
/*int i;*/

/*3.C++嵌套注释*/
/* int i = 0;
 *//*
 */

/* int i = 0;
  */int j = 0;


/* int i = 0;*//*int j = 0;
 */

/*
 *//*
 */int i =0;

/*4多重注释*/
/*
                                  int i;*/
/*5.字符串含有注释*/
"abcdefghijklmn  //   /*       *\~~~~!!!!!!!!"
```

```
//6.字符串
"aaaahfhkahkshfkshkahsfakf"
```