//定义PI的值

const float PI1=3.1415926536;

**帧的结构体**

typedef struct \_AudFrame

{

float fltFrame[FRM\_LEN];

float fltSte; //这一帧的短时能量

DWORD dwZcr; //这一帧的过零率

bool blVad;//判断这帧是否有效

struct \_AudFrame \* AudFrmNext;//下一帧的地址，不知道python有没有地址

}AudFrame;

**读取音频**

函数名：ReadWav(CString strFileName)

功能：读取音频

**预加重**

函数名：AudPreEmphasize(void)

功能：对所有采样点进行预处理

解释：

spSound：处理前的采样数据

fpPreSound：处理后的采样数据

fpPreSound[i] = (float)(spSound[i]) - float(spSound[i - 1]) \* 0.9375;

**分帧**

函数名：AudEnframe(float \*Sound,DWORD FrmLen,DWORD FrmSft,DWORD dwSoundLen)

功能：给每一帧的fltFrame[Frmlen]赋采样点的值，个数是帧长

解释：

Sound 输入的采样点数据的起始地址

Frmlen是帧的长度

FrmSft是帧移

dwSoundlen是采样点长度

**FrmNum表示有多少帧**

由采样点长度，帧长，帧移得到帧的个数FrmNum

FrmNum = (dwSoundLen - (FrmLen - FrmSft)) / FrmSft;

**汉明窗系数**

函数名：Hamming(DWORD FrmLen)

功能：求汉明窗系数，输入的是每一帧的帧长，要用到PI。这个数组是固定值，只有帧长决定

解释：

FrmLen 帧长，固定值

fltHamm[i] = (float)(0.54 - 0.46\*cos((2\*i\*PI1) / (FrmLen-1)));

**加窗**

函数名：AudHamming(DWORD FrmLen)

功能：输入的是每一帧的帧长，需要利用到求得的汉明窗系数，具体是每个采样点的值乘以汉明窗系数，再把结果赋予fltFrame[]

解释：

**需要先求得汉明窗系数**fltHamm[i];

(stpWav->fltFrame)[i] 每一帧采样点数据

for(DWORD i = 0; i < FrmLen; i++)

{

(stpWav->fltFrame)[i] \*= fltHamm[i];

}

**每一帧短时能量**

函数名：AudSte(fltSound \*fpFrmSnd, DWORD FrmLen)

功能：求每一帧的短时能量，即将所有这一帧的所有样点值相加，fpFrmSnd是帧第一个样本值

解释:

fltShortEnergy：每一帧短时能量

fpFrmSnd：每个样本点的值

for(int i = 0; i < FrmLen; i++)

{

fltShortEnergy += fabs(\*fpFrmSnd++);

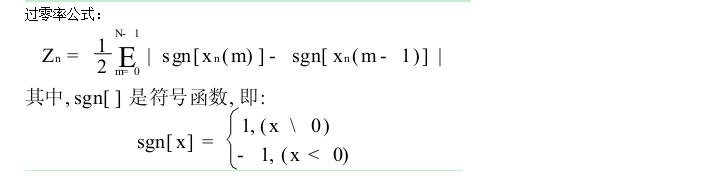
}

**一帧的过零率**

函数名：AudZcr(fltSound \*fpFrmSnd, DWORD FrmLen,fltSound ZcrThresh)

功能：求解一帧的过零率，fpFrmSnd帧第一个采样点地址，FrmLen帧长，ZcrThresh过零率阀值

解释：



fpFrmSnd样本点的值

DWORD CVad::AudZcr(fltSound \*fpFrmSnd, DWORD FrmLen,fltSound ZcrThresh)

{

DWORD dwZcrRate = 0;

for(int i = 0; i < FrmLen - 1; i++)

{

if((fpFrmSnd[i]\*fpFrmSnd[i + 1] < 0)&&(fabs(fpFrmSnd[i] - fpFrmSnd[i - 1]) > ZcrThresh))

dwZcrRate++;

}

return dwZcrRate;

}

**估计噪声阀值**

函数名： AudNoiseEstimate（）

功能：计算双门限阀值

解释：

fltSteThresh [2] 短时能量阀值，[0]高 [1]低

dwZcrThresh [2] 过零率阀值， [0]高 [1]低

ZcrThresh = 0;

StrThresh = 0.0;

**ZcrThresh = 所有帧的过零率之和**

**StrThresh = 所有帧的短时能量之和**

NoiseFrmLen 信号帧数

dwZcrThresh[0] = (float)(ZcrThresh) / NoiseFrmLen;

dwZcrThresh[1] = (float)(ZcrThresh) / NoiseFrmLen\*2.5;

fltSteThresh[0] = (float)StrThresh / NoiseFrmLen\*0.7;

fltSteThresh[1] = (float)(StrThresh / NoiseFrmLen)\*0.5;//\*0.95;

**端点检测**

**函数名:** AudVadEstimate(void)

功能：端点检测，需要用到估计阀值的函数，最后得出有效起始点和有效截止点

void CVad::AudVadEstimate(void) {

//Extract Threshold

DWORD ZcrLow=dwZcrThresh[1];

DWORD ZcrHigh=dwZcrThresh[0];

fltSound AmpLow=fltSteThresh[1];

fltSound AmpHigh=fltSteThresh[0];

WavStart=0;

WavEnd=0;

int end=0;

DWORD WordBeginFrm=0;

DWORD WordFrmCnt=0;

DWORD Silence=0; //Silence Length

DWORD frontSilence=0; //Silence Length

DWORD VoicedLength=0;//Voiced Word Length

DWORD VoicedFlag=1; //New Voiced Word Flag

DWORD status=0;

bool first=1;

int jamie\_silence=0;

int second=0;

AudFrame \*stpFirst=stpSoundFrm;

for(DWORD i=0;i<AudFrmNum-1;i++)

{

if(jamie\_silence>10)

{

first=1;

status=0;

VoicedFlag =0;

WordFrmCnt = 0;

jamie\_silence=0;

}

switch(status)

{

case 0:

case 1:

if ((stpFirst->fltSte)>AmpHigh)

{

VoicedFlag=2;

status=2;

if(first==1)

{

WordBeginFrm = i;

first=0;

}

if(i-WavStart>10 &&second==0)

{

second=1;

WavStart=0;

}

Silence=0;

WordFrmCnt++;

jamie\_silence++;

}

else if((stpFirst->fltSte)>=AmpLow && (stpFirst->dwZcr)>=ZcrLow)

{

status=1;

if(VoicedFlag==2)

{

WordFrmCnt++;

//Silence=Silence+1;

}

jamie\_silence++;

}

else if(Silence<=MAX\_SLIENCE\_LEN)

{

Silence++;

jamie\_silence++;

}

break;

case 2: //Speech Section

if((stpFirst->fltSte > AmpLow) || (stpFirst->dwZcr > ZcrLow))

{

if (WavStart==0)

{

if(stpFirst->AudFrmNext->AudFrmNext->AudFrmNext->fltSte<AmpHigh && stpFirst->AudFrmNext->AudFrmNext->AudFrmNext->dwZcr<ZcrLow)

{

status=0;

VoicedFlag =0;

WordFrmCnt = 0;

break;

}

WavStart=WordBeginFrm;

jamie\_silence=0;

}

WordFrmCnt = WordFrmCnt + 1;

//WavEnd=i-Silence;

}

else

{

Silence = Silence+1;

if (Silence < WORD\_MAX\_SLIENCE)

{

WordFrmCnt = WordFrmCnt + 1;

//WavEnd=i-Silence;

}

else if(WavEnd-WavStart< MIN\_WORD\_LEN)

{

status = 0;

VoicedFlag =0;

WordFrmCnt = 0;

WavStart=0;

first=1;

WordBeginFrm=0;

jamie\_silence=0;

}

else

{

//Get Voiced Word

VoicedFlag=0;

status = 0;

}

}

break;

default:

break;

}

vecFlt.push\_back(stpFirst->fltSte);

vecZcr.push\_back(stpFirst->dwZcr);

stpFirst=stpFirst->AudFrmNext;

}

for(int i = vecFlt.size()-1; i >= 0; i--)

{

if(vecFlt[i] > AmpLow && vecZcr[i] > ZcrHigh\*0.2 && i > 6 && vecFlt[i-6] > AmpHigh\*1.8)

{

WavEnd = i;

break;

}

}

dwWordLen=WavEnd-WavStart;

return;

}