Engenharia da computação – 1º período
Fundamentos de Programação 1 – 2017.1
Trabalho 2

Alunos: Vitor da Costa Mamede Corrêa & Gabriel Maia Gondim

RAs: 1905112 & 1936352

Todas as funções foram feitas presencialmente pelos dois membros da equipe, assim como os comentários; ficou a cargo do aluno Gabriel realizar uma série de testes para comprovar a eficiência das funções; e este relatório, a cargo do aluno Vitor; a revisão final do trabalho foi feita individualmente pelos dois integrantes do grupo, e as devidas correções foram feitas caso necessárias.

Todas as funções envolvem apenas manipulações simples com o vetor de dados e foram implementadas de maneira relativamente simples, nosso maior desafio foi na criação do algoritmo para a função 5, em específico na lógica que deveria ser utilizada para atenuar o sinal de acordo com o valor que extrapolou o limite estipulado e o parâmetro "n_passos".

Como as funções 1,2,3 e 5 recebem como parâmetro um ponteiro para o vetor de dados e seu tamanho, as modificações feitas por elas são feitas diretamente no vetor original, não sendo necessário a utilização de um segundo vetor para tais alterações.

A primeira função "mudaGanho" apenas multiplica todos os valores do vetor de dados original pelo valor do parâmetro "ganho", causando uma mudança na amplitude do sinal original, reduzindo-a caso o valor absoluto do parâmetro ganho seja menor que 1 ou aumentando-a caso seja maior que 1, reduzindo ou aumentando o "volume" do sinal audível, respectivamente, e invertendo a fase do sinal caso esse parâmetro seja negativo.

A segunda função "ruidoPeriodico" altera um valor do vetor de dados periodicamente para 1 ou -1, alternadamente, com um espaçamento de "intervalo" entre eles, onde "intervalo" é um parâmetro passado para a função. Ao alterarmos o valor de determinado ponto para 1 ou -1, criamos um súbito pico (positivo ou negativo) no sinal original e temos o efeito auditivo de um "estalo".

Para a terceira função foi criada uma função auxiliar "mediana" que recebe como parâmetro 3 variáveis do tipo *double* e retorna a sua mediana, utilizando um método que exclui o maior valor entre os 3 recebidos e depois retorna o maior valor entre os 2 menores, ou seja, o valor médio entre eles, com o auxílio de dois macros: "MAX(a,b)" e "MAX3(a,b,c)", que retornam o valor máximo de 2 e 3 entradas, respectivamente.

A terceira função “removeRuido” substitui todos os valores do vetor de dados pela mediana dos valores de determinada posição do vetor, da posição anterior e da posição posterior, com o auxílio de uma variável para que a mediana seja calculada sobre os valores do vetor original, e não do vetor modificado. Essa função funciona como um filtro, pois, supondo que o espaçamento entre dois picos seja de no mínimo 2 posições, três posições consecutivas terão um ou nenhum pico, que independente de ser positivo ou negativo, nunca será a mediana, justamente por ser um pico, portanto a saída da função que teve como entrada um sinal com picos periódicos (como o sinal de saída da função 3) será um sinal “limpo”, sem os “estalos” periódicos.

A quarta função “contaSaturacoes” faz uma simples varredura em todos os valores do vetor de dados e com o uso de uma condicional soma 1 a uma variável auxiliar para cada valor que não está dentro do limite estipulado “[-1,1]”, e ao final da varredura retorna esse valor, a quantidade de pontos saturados do sinal.

A quinta função “limitaSinal” deve funcionar como um limiter simples, para isso ela recebe como entrada um vetor com valores que extrapolam o limite “[-1,1]”. Quando o valor de uma posição ultrapassa o limite, esse valor é atenuado para 1 ou -1 (dependendo do valor original) e os valores das posições adjacentes são reduzidos proporcionalmente afim de manter a semelhança com o sinal original. Os artifícios utilizados para a implementação da função são:

- Uma variável “ratio” que recebe o módulo do inverso multiplicativo do valor de uma posição que extrapola o limite, que é utilizado para multiplicar esse mesmo valor e torna-lo 1 (ou -1)

- Uma variável “inc” que recebe o incremento que deve ser feito para reduzir proporcionalmente o valor das posições adjacentes, calculada a partir de ‘1-ratio’ (quanto “falta” para “ratio” atingir 1, o neutro multiplicativo, que quando usado na multiplicação não altera o valor multiplicado) dividido por “n_passos+1”, o número de posições adjacentes que devem ser atenuadas (o +1 é somado para que a primeira posição, atenuada para 1 ou -1, seja incluída na conta do incremento)

- Um segundo loop com o contador de iterações ‘j’ para atenuar os valores das posições adjacentes de uma determinada posição ‘i’, utilizando os índices [i+j] e [i-j] no vetor de dados, que a cada iteração incrementa ‘j’ em 1 e a variável “ratio” em “inc”.