Resumo do Projeto Prático 2

Vinícius Schnaider Zolet, Wilson Felipp dos Santos,
Pedro Henrique Belotto Frankiewicz

Departamento Acadêmico de Informática - DAINF
Universidade Tecnológica Federal do Paraná – Curitiba, PR – Brasil

Resumo: *Este artigo demonstra a participação de cada membro , os principais problemas encontrados e como eles foram superados para a elaboração dos programas solicitados no projeto prático 2 e também detalhes adicionais sobre os programas, de modo a explicar conceitos ou métodos que não ficaram explícitos nos comentários do programa.*

1-

Primeira Questão:

Para a primeira questão, o grupo se reuniu para discutir a lógica, onde percebendo que o calendário Discordiano tem a mesma quantia de dias e também tem um ano bissexto igual ao gregoriano, a ideia geral foi transformar o total de dias desde o começo do ano até o dia do mês inserido e transformar esse total para o Discordiano. Todos foram contribuindo para com o código e ideia de implementação até eventualmente chegar no resultado final.

Nessa ideia, chegamos no único problema de que durante o ano bissexto não tem um dia a mais, mas sim um dia entre o dia 59 e o 60, não alterando a ordem de dias para 74 no primeiro mês, o que acabava fazendo com o que os dias a partir do 60 sendo imprimidos errados, como 1 dia a mais.

Primeiro foi tentado tirar um dia da variavel de dias na impressao do ano bissexto, mas acaba dando errado quando ela dava como dia 1 (o que deveria ser o dia 73 do mes anterior), ficando como dia 0.

A solução para o problema foi para caso o ano fosse bissexto, dentro da função de impressão ele iria imprimir o dia de São Tiby no caso de dar 60. Se for qualquer mês depois do Caos ou se for acima do 60 (pois se for no mes do Caos, precisará ser feito também) ele irá tirar 1 dia ainda dentro da função. Se bater novamente 0 ele irá jogar para o dia 73 e retirar 1 mês, igual foi feito na função main, para não sair do padrão. Foi colocado o return para poder parar a função e não imprimir novamente o dia 60 do Caos, que seria errado.

2-

Segunda Questão:

Nessa questão o grupo foi reunido para discutir principalmente sobre como poderia fazer para termos a maior eficiência possível com um código não extenso. Ainda que cada um tenha contribuído com a parte do outro também, as funções principais ficaram: Vinicius fez a pesquisa sobre a biblioteca string.h, enquanto Wilson fez o projeto lógico do programa e Pedro fez a parte da transcrição com a string.h e a tradução do projeto lógico para o programa efetivo.

A primeira dificuldade que encontramos foi associar cada letra maiúscula do codigo (de A a U), que será dada na entrada pelo usuário , aos códons, que já estão tabelados sem gerar um if e else if para cada uma das letras. Para resolver esse problema, foi criado uma função, dentro dessa função foi criada uma matriz de dimensões 21x22, em que as linhas correspondem aos códons de A a U. As informações de entrada da função são dois vetores, o primeiro vetor sendo o código, e um vetor auxiliar que ira receber a tradução das letras para os códons.

Para realizar a tradução acessando essas linhas descritas acima, primeiro foi criado um loop que está limitado por strlen, queé o número do comprimento da string codigo, ignorando o "\0" do final, percorrendo assim toda sua extensão de letras. Para associar a letra maiúscula à linha correspondente da matriz, usou-se a tabela ASCII , onde A começa em 65, sendo assim [i-65]. Dentro do looping, é usado a função strcpy, em que o primeiro parâmetro é o vetor auxiliar que ira receber as strings dos códons, e o segundo parâmetro é a linha correspondente da matriz.

Para encontrar se a sequencia fornecida pelo usuário está dentro do código traduzido, foi criada outra função, e os parâmetros de entrada da função são dois vetores, um sendo o vetor com a string traduzida pela função descrita anteriormente, e o outro vetor é a sequencia. Para isso foi criado outro looping com limite sendo strlen e dentro deste foi utilizado strstr(), que procura uma string especifica dentro de outra string, e em seguida puxa o começo da nova busca para a posição final da sequencia encontrada, gerando um contador.

3-

Terceira Questão:

Na terceira questão o grupo se reuniu para entender primeiramente o que era necessario ser feito em cada uma das funções, o que foi também nosso maior desafio. Quando finalmente foi entendido o que o enunciado realmente pedia, que é uma função que manuseie os parâmetros pré-estipulados, foi quando começamos a trabalhar em conjunto para criar a lógia e, em seguida, traduzir da lógica para o código.

Em regra geral, todas as funções usam um looping que fica limitado por "n_amostras", ou "n_amostras-1" como é o caso da função 3, que é o tamanho do vetor "dados". Para facilidade será referido à este looping por "LoopingNA".

3.1- Para a função 1 foi apenas criado um único LoopingNA, que faz com que cada valor do vetor "dados" seja multiplicada pelo valor "ganho", seja ele qual for, gerando assim o aumento ou a diminuição do volume, de acordo com o valor colocado ao chamar a função.

3. 2.- Para a função 2 foi criado um único LoopingNA, que irá verificar se a posição atual do vetor se encontra na posição inicial ou nas posições dadas por intervalo-1, conforme pede o enunciado. Para tal, existem dois condicionais: um que ira verificar se o resto da divisao da posicao pelo intervalo é 0 e se o resultado da divisão desses valores é um valor par, que incluira então a posição incial [0], e um que irá verificar se o resto da divisão pelo intervalo é 0 e se o resultado da divisão é impar. Se for par, irá criar um estalo de valor 1 e se for impar irá criar um estalo de valor -1 nessas posições.

3.3- Para a função 3 o maior problema foi em entender como funcionava o *filtro da mediana**. Foi encontrado a solução do problema após pesquisa de ambos os integrantes para enfim entender que não apenas a posição atual poderia ser a menor das 3, mas todas as 3 tem essa chance. O erro estava em inicialmente testar apenas a posição atual como menor, ignorando as posições vizinhas. O segundo problema foi a necessidade de que fosse percorrido todo o vetor antes de ser substituído pela mediana, pois se fosse substituído imediatamente ele não iria gerar uma resposta correta na transcrição do áudio. Neste caso foi necessario criar um novo vetor, que foi denominado de "mediana" e usado alocação dinamica com tamanho "n_amostras" para deixar ela do mesmo tamanho de "dados", uma vez que ele servirá como auxiliar direta dele.

Neste caso, foram criados dois LoopingNA: O primeiro contém 3 condicionais, que irá verificar qual dentre os valores da posição atual e seus vizinhos é o menor e, dentro deles, 2 outros condicionais que verifica qual deste 2 maiores restantes é o menor, descobrindo assim a mediana e copiando ela para a poisção atual do vetor "mediana", criado anteriormente. O segundo contém apenas a transcrição dos dados das posições do vetor "mediana" para o vetor "dados" em suas respectivas posições, criando então um som sem os estalos.

*: Este método consiste em verificar os valores medianos de cada posição e ordenar de acordo com as mesmas. No caso de eliminação de estalos esse método funciona pois os valores 1 ou -1 nunca serão a mediana, o que fará com que aos poucos esses valores desapareçam do vetor, gerando um som limpo, ainda que esteja apenas levemente desincronizado com o som original utilizado para a criação do som com ruídos.