Aluno: Lucas Yukio Fukuda Matsumoto **Matrícula UTFPR:** 2516977

CSF 13 (Fundamentos de Programação 1) – 2022-2 – Relatório do Trabalho 1

Desafios da função 1:

- Como descobrir se o disparo atingiu o alvo?

Sendo os disparos e alvos circulares, e sabendo a posição do centro de cada um, imaginei como ficariam dispostos se o disparo errasse o alvo. Concluí que para errar, a distância entre os centros seria maior do que a soma de ambos os raios; se fosse igual a essa soma, teriam um ponto em comum e seria um acerto; se fosse menor que a soma, teriam uma área em comum. Essa ideia foi implementada na estrutura condicional.

- A distância entre os centros?

Foi encontrada por trigonometria, sendo a hipotenusa de um triângulo com catetos x do centro do disparo menos x do centro do alvo e y do centro do disparo menos y do centro do alvo.

- Qual o tipo de acerto?

Pensando nessa distância entre os centros, imaginando que um círculo está totalmente no interior de outro círculo, então a distância entre os centros somada ao raio do círculo menor não deve ser maior do que o raio do círculo maior. Essa ideia foi implementada para o caso se ocorreu acerto e o círculo maior for o alvo e o menor, o disparo, então foi um disparo coberto, e se foi ao contrário, então foi um alvo coberto, e se não foi nenhum desses casos, acertou comum.

Desafios da função 2:

- O enunciado pede para retornar a soma total de quantos alvos foram atingidos por cada disparo, sem eliminar as repetições ou pede para retornar só a soma de quantos alvos foram atingidos, sem contar um mesmo alvo duas vezes?

Fiquei confuso nesse trecho do enunciado, acho que ficou ambíguo, pois interpretei que não era para contar o mesmo alvo duas vezes, mas desse jeito não faria sentido usar a função 1 na função 2, e também não passaria nos testes disponibilizados, então contei todos os acertos, sem eliminar os acertos repetidos,

para fazer sentido chamar a função 1 e seguir somando os valores retornados por ela para chegar ao total de acertos.

Desafios da função 3:

- Quantos bits a sequência completa possui?

Resolvi utilizar o sizeof(), que retorna a quantidade de bytes do tipo da variável. Bytes possuem 8 bits, então multiplica-se por 8. Achei que fica mais claro para o leitor do que jogar o valor 16, pois alguém pode esquecer quantos bits tem ou mudar o tipo.

- Como enumerar cada bit do campo?

Enumerei em ordem decrescente de “nbits”, número de bits do campo, até 1, da esquerda para a direita, por ficar mais parecido com a ordem decrescente dos expoentes de 2 para a conversão para decimal.

- Como eliminar os outros bits e só olhar para o bit que quero analisar?

Fiz uma cópia completa da sequência de bits para poder realizar operações bit-a-bit, para extrair um bit sem perder a informação dos outros bits. Para analisar um determinado bit “i” do campo, desloco a sequência para a esquerda (número de bits antes do campo + número de bits do campo – número do bit “i” do campo) até que o primeiro bit dela seja o bit “i”. Depois, desloco a sequência para a direita (total de bits da sequência – 1 bit) até que o último bit seja o bit “i”. Os bits que ultrapassam a capacidade de armazenamento da variável são perdidos, por isso resta apenas o bit desejado. Se for 0 em binário, o valor da variável em decimal também é 0, e se for 1 em binário, será 1 também em decimal.

- Como converter de binário para decimal?

É feito pela soma de potências de 2. Cada bit 1 representa a presença de uma determinada potência de 2, iniciando com expoente 0 no primeiro bit à direita e acrescendo 1 a cada bit da direita para a esquerda. Um bit 0 indica ausência daquela potência de 2, e não fará nada. Como enumerei os bits “i” de “nbits” até 1, da esquerda para a direita, então, cada vez que o valor da variável é 1, significa que o bit “i” é 1, e a função irá calcular a potência de 2 elevado a “i”-1, pois o expoente irá de 0 a “nbits”-1, e adicionará à variável decimal. Se “i”-1=0, apenas somará 1, que é 2 elevado a 0, à variável decimal. O resultado retornado será o valor do campo em decimal.