Documentação Programação – Processo Seletivo CrossBots 2022/2

muito_simples_p1.cpp

Esse exercício não tive muita dificuldade, pois a distância de um vetor é calculada através da fórmula abaixo:

$$|v| = \sqrt{x^2 + y^2}$$

(Fonte: https://mundoeducacao.uol.com.br/matematica/angulo-entre-dois-vetores.htm)

Então, apenas transcrevi a fórmula do módulo utilizando as funções sqrt() e pow() da biblioteca math.h.

A única dificuldade que eu tive, foi que quando digitei o símbolo de menos (-), o compilador entendeu que o símbolo não era de menos e sim, o símbolo de hífen. Só que tanto no Visual Studio, quanto no CodeBlocks, o erro informada não era muito intuitivo, aí pesquisando na internet o erro, descobri o motivo do erro e consegui compilar o código corretamente.

Variável	O que armazena
int p1[2]	Coordenadas do primeiro ponto.
int p2[2]	Coordenadas do segundo ponto.
float d	Armazena o cálculo do módulo do vetor formado por p1 e p2.

muito_simples_p2.cpp

Esse exercício também não tive grandes dificuldades, bastando apenas pesquisar a fórmula de conversão de graus Fahrenheit para Celsius e aplicá-la no código, com os parênteses corretamente.

O único problema foi que eu não sabia como limitar 2 casas decimais no float em C++, pois não me recordava de como fazia isso. Na verdade, só lembro de ter visto isso em C, e não em C++. Então, após uma pesquisa na internet, encontrei os métodos std::cout<<fixed() e std::cout<<precision() que limita os valores impressos na tela de acordo o valor informado em std::cout<<precision().

Variável	O que armazena
float celsius	Armazena o valor recebido em graus Celsius.
float temp	Armazena o cálculo da conversão de graus Fahrenheit em graus Celsius.

simples_p1.cpp

Esse também achei fácil, até que achei estranho ter saído tão facilmente que resolvi ampliar o que foi pedido e coloquei uma limitação para prevenir que

fosse digitado valores negativos e/ou valores maiores que o tamanho do array que contém os valores a serem lidos.

Variável	O que armazena
int lista[]	Armazena a lista de valores a ser analisado.
int x	Armazena o valor x para procurar na lista.
int y	Armazena o valor y para procurar na lista.
int soma	Calcula e armazena a soma dos elementos da posição x e y da lista.
bool flag	Flag para checar se o valor digitado é negativo e/ou maior do que o tamanho da lista.

simples_p2.cpp

Esse exercício eu vi recentemente nas aulas de programação, então foi bem simples de resolver, bastando apenas checar os divisores do número digitado pelo usuário e somando cada divisor encontrado numa variável chamada “soma”. Após isso, no final do loop, checa se o valor da soma é igual a variável digitada pelo usuário. Se for, ele imprime o valor booleano 1 na tela, se não, imprimi o valor booleano 0.

Variável	O que armazena
int num	Variável para armazenar o valor que será analisado se é perfeito ou não.
bool flag	Apenas para checar se o valor digitado é negativo ou não.
int soma	Armazena o valor da soma dos divisores do número para comparar com o núm.

simples_p3.cpp

Esse exercício tomou um pouco mais de tempo, pois eu sabia que iria precisar utilizar os conhecimentos que eu vi em “Estrutura de Dados” enquanto eu cursava Jogos Digitais, aí eu lembrei que isso poderia ser resolvido utilizando um dos conceitos dessa matéria. Aí pesquisando na internet, relembrei alguns conteúdos e vi que poderia utilizar a biblioteca list para esse caso, pois eu poderia remover qualquer item da lista, a qualquer momento. E foi o que fiz, criei uma lista, inicializei ela com alguns valores e na função “encontra_Primo()" checamos valor a valor da lista. Como a lista precisa de um iterator para se movimentar por ela, precisei checar se o elemento analisado é 1, se fosse, já era removido. E caso ele estivesse na primeira posição da lista, não precisa retornar o iterator uma posição anterior, pois quando um item é removido da lista, o próximo item da lista, fica na posição do item removido, e como precisamos analisar esse número também, precisamos retornar o iterator uma posição para trás e continuar nossa análise.

Aí foi criado um loop para checar se o número é primo ou não, e caso o número da lista tivesse um divisor maior ou igual a 2 ou menor que o número analisado, então, ele já não é primo, pois foi encontrado um terceiro divisor

além do 1 e dele mesmo, então, já removemos ele da lista, da mesma forma que é feito caso o número fosse 1.

Variável	O que armazena
list<int> lista	Uma lista que armazena os valores a serem analisados, que precisa de um list<int>::iterator para analisar cada membro dela.

simples_p4.cpp

Esse exercício também foi simples de resolver, eu apenas não me recordava de como lia cada caractere da string, até que lembrei que pesquisei na internet e lembrei que tem como utilizar o “for(char i : frase)” para checar cada char da string até a mesma finalizar.

Variável	O que armazena
string frase_analisar	Frase a ser analisada.
char letra_analisar	Armazena o char a ser analisado.
int contador	Contador de quantas vezes foi repetido o char na string.

intermediarios_p1.cpp

Esse exercício, deu um trabalhinho, pois tive que pesquisar como é calculado o ângulo entre 2 vetores e relembrei das aulas de GA. Aí eu montei os cálculos necessários para achar os 2 vetores que formam o ângulo através dos pontos e utilizei a fórmula abaixo para calcular o cosseno do ângulo:

$$cos\varphi = \frac{x_1.x_2 + y_1.y_2}{\sqrt{{x_1}^2 + {y_1}^2}.\sqrt{{x_2}^2 + {y_2}^2}}$$

(Fonte: https://mundoeducacao.uol.com.br/matematica/angulo-entre-dois-vetores.htm)

Só que, inicialmente, eu fiz a fórmula do cosseno do ângulo e do arcocosseno do ângulo, tudo em uma mesma fórmula, utilizando apenas as variáveis dos vetores que criei através do cálculo, ou seja, deu uma confusão de parênteses, uma linha extremamente grande, cheios de valores, pows, e o resultado estava dando Nan toda hora.

Então, após muita análise do código e das fórmulas, verifiquei que estava calculando os vetores de forma errada, pois eu estava calculando os vetores AB e BC, em vez de calcular os vetores BA e BC. Corrigi isso, mas ainda estava dando Nan no cálculo do ângulo.

Comecei a separar cada parte da fórmula do cálculo do cosseno do ângulo em variáveis separadas e ver em qual parte da fórmula estava ocorrendo o Nan. Logo, descobri que o erro, estava no cálculo do cosseno, pois não coloquei parênteses na multiplicação dos módulos, ou seja, ele estava dividindo o produto interno dos vetores, com o módulo do primeiro vetor e depois multiplicando o resultado pelo módulo do segundo vetor. Com isso, o cosseno estava dando valores acima de 1, que estava resultando no Nan. Após essa correção, foi

calculado o cosseno corretamente e, consequentemente, o arcocosseno foi calculado corretamente, foi feita a conversão de radianos para graus e retornou o valor do ângulo corretamente.

Variável	O que armazena
int p1[2]	Variável para armazenar a posição x e y do ponto 1.
int p2[2]	Variável para armazenar a posição x e y do ponto 2.
Int p3[2]	Variável para armazenar a posição x e y do ponto 3.
double angulo	Armazena o valor resultante do arcocosseno da variável cos_angulo e convertida em graus.
double vetor1[2]	Armazena as coordenadas do vetor BA.
double vetor2[2]	Armazena as coordenadas do vetor BC.
double modulo_vetor1	Armazena o cálculo do módulo do vetor BA.
double modulo_vetor2	Armazena o cálculo do módulo do vetor BC.
double produto_interno	Calcula e armazena o valor do produto interno das coordenadas do vetor1 e do vetor2.
double cos_angulo	Calcula e armazena o valor obtido pela divisão do produto interno dos vetores vetor1 e vetor2, pela multiplicação dos módulos armazenado em modulo_vetor1 e modulo_vetor2.

intermediarios_p2.cpp

Esse exercício eu confesso que não fiz da melhor forma possível, pois eu utilizei a biblioteca list novamente, e utilizei o comando sort() e unique() para remover os itens repetidos dela, onde o comando sort() organiza os números da lista em ordem crescente e o comando unique() só remove os números repetidos caso eles sejam seguidos um após o outro. Eu sei que conseguiria fazer um programa que remove os números repetidos sem precisar ordená-los, utilizando conteúdos de Estrutura de Dados que eu lembro de ter visto exercícios do tipo há anos, quando fiz a matéria pela primeira vez. Mas, como no exercício não informa que precisa ser ordenado ou não, resolvi deixar essa solução mesmo, pois fiz o que foi pedido, de forma rápida e simples.

Variável	O que armazena
list<int> lista	Armazena a lista dos números a ser verificado e removido os repetidos.

difíceis_p1.cpp

Nesse exercício, acho que a melhor forma é utilizando a biblioteca vector, pois ela tem uma gama maior de comandos para ter um controle melhor de quem será removido e de quando for encontrado um diamante. Não cheguei a terminar o exercício, mas se eu tivesse um pouco mais de tempo livre, eu sei que conseguiria finalizar a lógica que iniciei. Mas até onde eu fiz, foi criar um vector de strings, que irá receber as n strings para serem realizadas os testes para

encontrar os diamantes. Após isso, iria para uma função procurarDiamantes(), onde iria passar por cada string do vetor e verificar char por char, se localizou o início do diamante, e se após localizar o início do diamante, ele irá localizar o final do diamante. Se sim, iria remover todo o conteúdo do vetor desde o valor da posição do início do diamante, até a posição do final do diamante, acrescido de 1, pois a função erase() não inclui o último elemento.

Após isso, precisaria verificar qual posição da string seria checada para consertar o código ou mudar alguma lógica, partes que não deram tempo de finalizar. E não consegui compilar o código, pois não me recordava como compara o conteúdo da string dentro de um vetor num for, para checar char por char das strings dentro do vetor. Nada que uma pesquisa não resolvesse, mas está quase no tempo limite da entrega, então, entregar e documentar até onde consegui fazer.

Variável	O que armazena
int n	Valor de casos de testes a serem verificados.
string mina	String que será recebido os casos de testes com possíveis diamantes e areia.
vector<string> testes	Vetor que irá receber todas as n strings inseridas pelo usuário.
int diamantes_encontrados	Irá somar quantos diamantes foram encontrados.
int pos_inicio_diam	Armazena a posição do início de um possível diamante.
int pos_final_diam	Armazena a posição do final de um diamante.
bool inicio_diamante	Flag para checar se já achou o início de um diamante.
bool fim_diamante	Flag que irá checar se chegou no final do diamante, para poder remover o diamante encontrado.

dificeis_p2.cpp

Não deu tempo para começar a fazer, mas cheguei a ler e compreender o problema, e de todos os problemas, esse seria o mais difícil de fazer, mas não por nível de dificuldade, mas sim, pela complexidade em criar o código, a análise de cada tabuleiro, criar a movimentação das peças, então, seria um código bem massivo na fase de criação dele, para então, resolver o problema. Queria ter iniciado, mas infelizmente, não deu.