Documentação: Trabalho Problemas Combinatórios, Matemática Discreta

Soma Máxima de um Subvetor

O programa inicia solicitando a entrada que deverá ser um número N, 3 =< N <= 20, caso seja fornecida uma que não se encontra nesse intervalo pré-determinado o programa acusará uma mensagem de erro " Entrada Inválida", e outra dizendo "Digite o tamanho do seu vetor novamente" entrando em um looping infinito até que a entrada digitada seja válida, como mostra essa parte do programa.

```
/*
do      {
        scanf("%d", &N);
        if(N < 3 || N > 20)
        {
                printf("Entrada Invalida!\n");
                printf("Digite o tamanho do seu vetor novamente:\n");
        }
        } while(!(N >= 3 && N <= 20));
*/
```

Em seguida, é criado um vetor (quantNumeros) que será inicializado com N (N = valor da entrada fornecida ao programa) posições, e foi criado um laço com um scanf, para que seja lido um valor e em seguida esse valor é inserido em uma das N posições do vetor. (Como mostra o seguinte trecho do programa)

```
/*
for (i = 0; i < N; ++i)
        {
                scanf("%d", &quantNumeros[i]);
        }
*/
```

Após isso o programa inicia seu fluxo principal de operações, foi criado um laço mais externo que itera pelos N tamanhos diferentes de subvetores exemplo: Se a entrada for N = 3, ele varrerá vetores de tamanho 1, 2, e 3. O outro laço que roda dentro do anterior identifica como será analisado um

subvetor 1 em 1, 2 em 2... N em N posições e identifica a posição inicial do subvetor. Na sequência dentro do laço anterior identificando a posição inicial do subvetor lida anteriormente e realiza a soma de cada posição do subvetor, e nesse laço tem uma condicional que faz uma comparação de resultados guardados na variável soma (inicializada = 0) de cada iteração com a variável maior, caso (soma > maior) faça (maior = soma) e guarde os índices caso contrário descarte a soma desse subvetor. Ao final das N iterações o programa imprime o valor da variável **maior** que contém o maior resultado da soma das posições do subvetores. (Parte do programa que faz o que foi citado acima)

```
/*
for (i = 0; i < N; ++i
{
        for (j = 0; j < (N-i); ++j)
        {
                soma = 0
                        for (k = 0; k < (i+1); ++k)
                        {
                                soma += quantNumeros[j+k];
                                if (soma > maior)
                                {
                                        maior = soma;
                                        indices[0] = j;
                                        indices[1] = j+k;
                                }
                        }
        }
}
        printf("Soma: %d\n", maior);
*/
```

Finalmente, a última parte do programa contém apenas uma condicional que avalia se o valor salvo na variável **maior** é maior que 0 para imprimir os indicies caso contrário imprima 0 (trecho do programa).

```
/*
if (maior > 0)
      {
         printf("Indices: %d a %d\n", indices[0]+1, indices[1]+1);
      }
      else printf("Indices: %d a %d\n", indices[0], indices[1]);

*/
```

Quadrado Mágico

Um quadrado mágico de ordem N é um arranjo de N^2 números, geralmente inteiros distintos, em um quadrado, de modo que os N números em todas as linhas, todas as colunas, e ambas as diagonais somam a mesma constante. Um quadrado mágico contém os números inteiros de 1 a N^2.

A soma constante em cada linha, coluna e diagonal é chamada de constante mágica ou magia, M. A constante mágica de um quadrado mágico normal depende apenas de N e tem o seguinte valor:

Constante Mágica = M = N (N^2 + 1) / 2.

A lógica do programa para a Matriz 4x4 (quadrado mágico par duplo) é preencher a diagonal principal da matriz utilizando variável **contador** e ir incrementando (contador++) juntamente com a diagonal secundária, na sequência é preenchido as colunas subindo de trás para a frente e da direita para a esquerda com auxílio da variável **contador2** decrementando ela com **contador2--** (contando de 17 até 1).

```
        */
        for(colunas = 0; colunas < N; colunas++)
              {
                 for(linhas = 0; linhas < N; linhas++) // laço que realiza o preenchimento da
matriz de acordo com as condições abaixo.
                {
                   contador++;
                   contador2--;
                      if(linhas == colunas) // preenche diagonal principal
                      {
                         MATRIZ[linhas][colunas] = contador;
                      }
                         else if(linhas == (N-1-colunas)) // preenche diagonal secundária
                         {
                            MATRIZ[linhas][colunas] = contador;
                         }
                            else if(MATRIZ[linhas][colunas] == 0) // preenche o restante
das posições
                            {
                               MATRIZ[linhas][colunas] = contador2;
                            }
                }
             }
          }
         /*
```

Na sequência caso a entrada do programa seja um número ímpar ele executará o trecho do programa relacionado a Matriz ímpar (Quadrado mágico 3 ou 5), a lógica dessa parte do fluxo de execução é dividida em 4 etapas: Primeira etapa, é alocar o número 1 na linha 3 coluna 2. (Trecho correspondente)

```
        /*
i = N-1; j = N/2; // inicializando as variáveis (i) e (j) para começar a matriz com o (1) na posição correta.

   for(k = 0; k < N*N; k++) // passo 4, inserir os N números, repetir até que toda a matriz esteja completa

      MATRIZ[i][j] = num; // passo 1, insere o número 1 na posição correta da matriz.

   1 na linha 3 coluna 2

        */
```

Segunda Etapa a partir da posição ***i = N-1; j = N/2;*** onde foi colocado o número 1 o fluxo deverá seguir da seguinte maneira uma posição para esquerda e uma para baixo, sempre guardando a posição anterior, essa parte aparece no trecho a seguir do programa

```
/*

linha_ant = i; coluna_ant = j; // a cada iteração a posição anterior é guardada.

      j--; i++; // passo 2, percorrer a matriz uma posição a esquerda e uma para baixo

      if (i > N-1)

      {

         i = 0;

      }

      If (j < 0)

      {

         j = N-1;

      }
*/
```

Passo 3, é analisada uma condicional relacionada ao passo 2, caso a posição encontrada após realizar o passo 2 seja um número diferente de 0, ele deve retornar à posição anterior e subir uma posição. (Parte do programa que isso ocorre)

*/

If (MATRIZ[i][j] != 0) *//laço: passo 3, se for encontrada uma posição com número diferente de 0, voltar a posição anterior (antes da realização do passo 2) e subir uma posição.*

```
{
 i = linha_ant;
 j = coluna_ant;
 i--;
}
```

/*

Passo 4 é o **for** onde essas condições estão inseridas, ao fim de avaliar cada uma delas (N^2) a variável **num** é incrementada **num++** até toda a matriz estar preenchida. (Trecho correspondente)

/*

For (k = 0; k < N*N; k++) *// passo 4, inserir os N números, repetir até que toda a matriz esteja completa*

```
{
   (CONDICIONAIS A SEREM AVALIADAS)
   num++; // adiciona os números na matriz
}
```

*/

Por fim foi adicionada uma parte extra para verificação de resultados e imprimir o valor da Constante mágica de acordo com essa avaliação para os dois tipos de Quadrado Mágico. Esse trecho verifica linhas, colunas e diagonais guardando o valor da soma dos índices em uma variável **temp** que é igualada com a variável **somaMagica** e incrementa a variável **acertos** por fim esta é comparada com a expressão (2*N) +2 que corresponde ao número total de linhas, colunas e diagonais caso seja verdadeira o programa passou no teste e é impresso o valor da soma através da variável **somaMagica.** (Trecho do programa correspondente)

*/

```
>>Verificação de Resultados<<
Esta parte do programa apenas realiza comparações para efeito
de confirmação de que todas linhas, colunas e diagonais possuem
```

```
    o mesmo resultado quando somadas as componentes de cada
    posição (i)(j) da matriz.

printf("Matriz: %d x %d \n\n",N, N);

for(i = 0; i < N; i++)  // imprime a matriz preenchida
{
   for(j = 0; j < N; j++)
      printf("%3d", MATRIZ[i][j]);
      printf("\n");
}
printf("\n");

for(i = 0; i < N; i++) // verifica linhas
{
   for(j = 0; j < N; j++)
   {
      temp += MATRIZ[i][j];
   }
   if(temp == somaMagica)
   {
      acertos++;
   }
   temp = 0;
}

for(i = 0; i < N; i++) // verifica colunas
{
   for(j = 0; j < N; j++)
   {
```

```
      temp += MATRIZ[j][i];
   }
   if(temp == somaMagica)
   {
       acertos++;
   }
   temp = 0;
}
for(i = 0; i < N; i++)  // verifica diagonal principal
{
   temp += MATRIZ[i][i];
}
if(temp == somaMagica)
{
    acertos++;
}
temp = 0;
for(i = 0; i < N; i++) // verifica diagonal secundária
{
   for(j = 0; j < N; j++)
   {
      if(i == N-1-j)
      {
         temp += MATRIZ[i][j];
      }
   }
      if(temp == somaMagica)
      {
         acertos++;
      }
```

```
}
if(acertos == (N*2)+2) // Mostra o resultado da Soma
{
   printf("Soma: %d\n", somaMagica);
}else printf("ERRO!\n");
```

Fim do fluxo de execução!

```
system("pause");
 return 0;
```