UNIVERSIDADE FEDERAL DO PAMPA - CAMPUS BAGÉ

ENGENHARIA QUÍMICA

ALGORITMOS E PROGRAMAÇÃO

Avaliação de Recuperação:
Relatório de Resolução dos Algoritmos

Maria Eduarda Soares Delgado

Bagé

2021

1)

Nesta questão, na função *"main"* foi utilizado um vetor que irá contar 4 elementos, onde por meio do laço de repetição *"for"* é requisitado ao usuário que seja inserido um valor que possua apenas um único dígito, este mesmo dígito único é assegurado pelo laço de repetição *"do-while"* que irá forçar com que o usuário insira um valor correto.

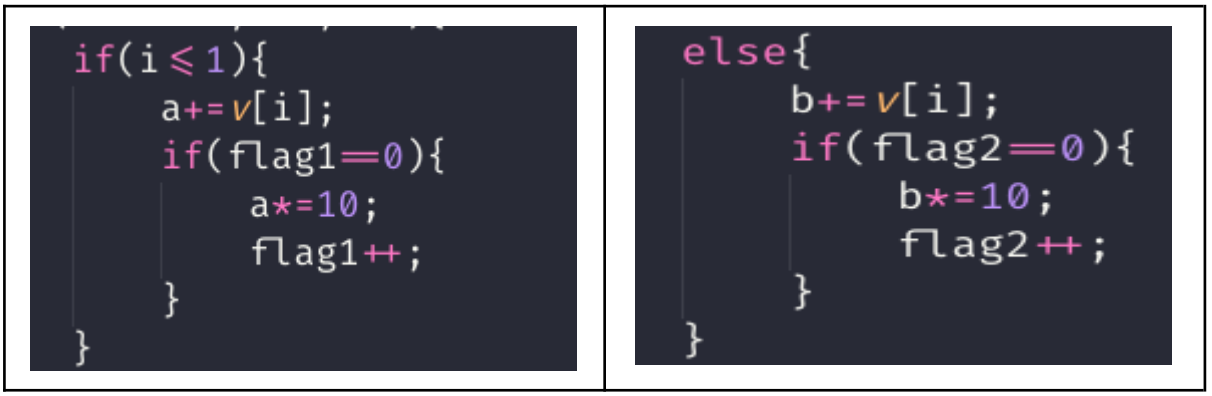
```
printf("\n---------Digite 4 Digitos, 1 por vez!!!---------\n");
for(int i=0;i<4;i++){
    do{
        printf("\nDigite o digito %i: ",i+1);
            scanf("%i", &num);
        if(num >= 10)
            printf("\nDigite apenas um digito!!!\n");
    }while(num >= 10);
    vet[i]=num;
}
```

A função *"temCaracteristica"* foi modificada do protótipo, sendo enviado o mesmo vetor onde foram inseridos os 4 dígitos únicos. A mesma função é "chamada" dentro como um argumento do "if", tendo em vista que ela retornará um ou zero, onde caso seja um será apresentado ao usuário a resposta de verdadeiro e caso seja zero será apresentado ao usuário a resposta de falso.

```
if(temCaracteristica(vet))
    printf("\nValor Possui a Caracteristica!!\n");
else
    printf("\nValor Não Possui a Caracteristica!!\n");
```

Na função *"temCaracteristica"*, as posições zero e um do vetor serão agrupados em uma única variável em unidade e dezena. Para a unidade o valor da posição zero apenas é adicionado a variável *"a"*, para que seja adicionada a unidade e para que seja feita a adição da dezena, o mesmo valor da variável *"a"* é multiplicado por 10(dez), que é assegurado pela variável *"flag1",* após isso é feita a soma da posição um do vetor.

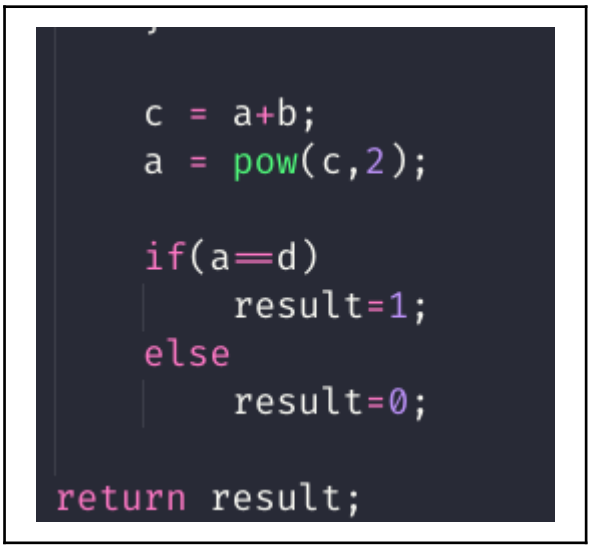
O mesmo processo é feito para as posições dois e três do vetor, porém com o uso das variáveis *"b"* e *"flag2"*. Além destas operações, na variável *"d"* é feita a adição de todas as posições do vetor para que seja verificado o valor inicial que o usuário digitou.



```
if(i<=1){
    a+=v[i];
    if(flag1==0){
        a*=10;
        flag1++;
    }
}
```

```
else{
    b+=v[i];
    if(flag2==0){
        b*=10;
        flag2++;
    }
}
```

Após isso, na variável *"c"* é realizado a soma de "a" e "b", e logo em seguida é realizada a potência. Caso o resultado da potência seja verdadeiro, será retornado o valor 1, caso não, zero.



```
    c = a+b;
    a = pow(c,2);

    if(a==d)
        result=1;
    else
        result=0;

return result;
```

2)

Nesta questão, na função *"main"* foram declaradas as variáveis vetores *"vet"* e *"vet2"*, após isso foram chamadas as devidas funções requisitadas na questão que serão descritas em ordem logo abaixo.

```
int vet[5];
int vet2[5];

insere(5,vet);
printf("\n------------Vetor Com Valores Aleatorios!!!------------\n");
show(5,vet);
ordena(5,vet);
printf("\n------------Vetor Com Valores Ordenados!!!------------\n");
show(5,vet);
int qtd=simplifica(5,vet,vet2);
printf("\n------------Vetor Sem Valores Repetidos!!!------------\n");
show(qtd,vet2);
```

A função *"insere"* recebe como parâmetros a quantidade de valores do vetor e a declaração do vetor. Nesta função existe apenas o laço de repetição *"for"* com objetivo de inserir valores aleatórios de zero até dez nas posições do vetor.

```
void insere(int n,int V[]){
    for(int i=0;i<5;i++){
        V[i] = (rand() % 10);
    }
}
```

A função *"show"* recebe como parâmetros a quantidade de valores do vetor e a declaração do vetor. Nesta função existe apenas o laço de repetição *"for"* com objetivo de apresentar o valor alocado em cada posição do vetor.

```c
void show(int n,int V[]){
    for(int i=0;i<n;i++){
        printf("\nVet[%i]: %i",i,V[i]);
    }
}
```

A função *"ordena"* possui dois laços de repetição *"for"* que estão um após o outro, com o objetivo de ordenar os valores. O primeiro *"for"* tem o objetivo de que seja repetido o número de valores igual ao de elementos do vetor. O segundo *"for"* tem o objetivo de ordenar os valores das posições do vetor.

```c
void ordena(int n,int V[]){
    for(int j=0;j<n;j++){
        for(int i=0;i<n-j-1;i++){
            if(V[i]>V[i+1]){
                int aux = V[i];
                V[i] = V[i+1];
                V[i+1] = aux;
            }
        }
    }
}
```

A função *"simplifica"* recebe como parâmetros a quantidade de valores que o vetor possui, o vetor e o vetor que não possuirá repetições. Além disso, possui dois laços de repetição *"for"* que estão um após o outro, com objetivo de verificar se um número possui mais de duas repetições, pois ele sempre irá verificar a si mesmo, onde a *"flag1"* será incrementada e caso haja mais de uma repetição(que é considerada o próprio valor), é considerado que há uma cópia do mesmo e é descartada.

Após isso, é retornado o variável *“cont”* que irá possuir a quantidade de valores que o vetor sem repetições possui.

```
int simplifica(int n,int V1[],int V2[]){

    int flag1,cont;
    flag1=cont=0;

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(V1[i]==V1[i+1])
                flag1++;
        }
        if(flag1<2){
            V2[cont]=V1[i];
            cont++;
        }
        flag1=0;
    }
return cont;
```

3)

A função *“main”* possuí a declaração de uma variáveis matriz de tipo double nomeada *“mat”* e um vetor denominado *“vet”* que será utilizado na função *“minmax”* para armazenar a linha e a coluna do elemento **MINMAX**. Após isso, existem dois laços de repetição *“for”* com o objetivo de inserir valores randômicos entre zero e noventa e nove. Finalizando, todas as funções criadas são chamadas.

```c
int main(void){

    srand(time(NULL));

    double mat[6][6];
    int vet[2];

    for(int i=0;i<6;i++){
        for(int j=0;j<6;j++){
            mat[i][j]=(rand() % 100);
        }
    }
    printf("\n------------APRESENTANDO MATRIZ!!!------------\n");
    show_mat(6,6,mat);
    double valor = minmax(6,6,mat,vet);
    printf("\nO elemento MINMAX: %.2f",valor);
    show_vet(2,vet);
return 0;
}
```

A função *“show_mat”*, recebe como parâmetros a quantidade de posições da matriz e a mesma. Dentro da função existem dois laços de repetição *“for”* , com o objetivo de apresentar os valores da matriz.

```c
void show_mat(int lin,int col,double M[lin][col]){
    for(int i=0;i<lin;i++){
        for(int j=0;j<col;j++){
            printf("%.2f\t",M[i][j]);
        }
        printf("\n");
    }
}
```

A função *“show_vet”*, recebe como parâmetros a quantidade de posições do vetor e o mesmo. Dentro da função é apresentado por meio de um *“printf”* o valor da linha e coluna do elemento **MINMAX**.

```c
void show_vet(int qtd,int V[qtd]){
    printf("\nA linha do elemento MINMAX: %i\t",V[0]);
    printf("\nA coluna do elemento MINMAX: %i\t",V[1]);
}
```

A função *“minmax”* recebe como parâmetros o número de linha e colunas, além da própria matriz e o vetor que irá receber a linha e coluna do elemento **MINMAX**. Dentro da função, existem dois laços de *“for”* que estão declarados um após o outro com o objetivo de encontrar o menor elemento e a sua linha, que serão armazenados nas variáveis *“min”* e *“linha”*. Após isso, existe um outro *“for”* com o objetivo de percorrer as colunas da linha que possui o menor valor da matriz, para que seja encontrado o maior valor da linha. Tendo encontrado o maior valor da linha, o valor e o número da coluna serão armazenados nas variáveis *“max”* e *“coluna”*. Finalizando, os valores das variáveis *“linha”* e *“coluna”* são armazenados nas posições zero e um, respectivamente, dentro do vetor enviado por parâmetro e é retornado o valor da variável *“max”*, que é o **MINMAX**.

```c
double minmax(int lin, int col,double M[lin][col],int V[]){

    double min=M[0][0];
    double max=M[0][0];
    int linha,coluna;

    for(int i=0;i<lin;i++){
        for(int j=0;j<col;j++){
            if(min>M[i][j]){
                min=M[i][j];
                linha=i;
            }
        }
    }
    printf("\nO menor elemento eh: %.2f",min);
    for(int j=0;j<col;j++){
        if(max<M[linha][j]){
            max=M[linha][j];
            coluna=j;
        }
    }
    V[0]=linha;
    V[1]=coluna;
return max;
```