144.Crie uma função que receba os coeficientes de uma função de 2º grau e retorne as

raízes. Utilize alocação dinâmica e ponteiro.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void raizes(int a,int b,int c,float\* raiz)

{

int delta;

delta=pow(b,2)-4\*a\*c;

if(delta>=0){

raiz[0]=(-b+sqrt(delta))/2\*a;

raiz[1]=(-b-sqrt(delta))/2\*a;

printf("raiz 1 : %.2f raiz 2 : %.2f",raiz[0],raiz[1]);

}

else{

printf("Não existem raízes reais.");

}

}

int main(void)

{

int a,b,c;

float \*raiz;

raiz=(int \*)malloc(2\*sizeof(int));

printf("digite a,b e c da função quadrática: ");

printf("a: ");

scanf("%d",&a);

printf("b: ");

scanf("%d",&b);

printf("c: ");

scanf("%d",&c);

raizes(a,b,c,raiz);

return 0;

}

145.Faça um programa que acha o maior e o menor inteiro dentro de um vetor de dez inteiros, calcule a soma e imprima a soma. Usar as variáveis a seguir:

int v[10], I, \*maior, \*menor;

#include <stdio.h>

#include <stdlib.h>

void soma\_maior\_menor(int v[10],int \* maior,int \*menor,int \* soma)

{

int maior2=v[0];

int menor2=v[0];

int i;

for(i=1;i<10;i++){

if(v[i]>=maior2){

maior2 = v[i];

}

if(v[i]<=menor2){

menor2 = v[i];

}

}

\*maior=maior2;

\*menor=menor2;

\*soma=maior2+menor2;

}

int main(void)

{

int maior;

int menor;

int v[10];

int soma;

int i;

for(i=0;i<10;i++){

printf("v[%d]: ",i);

scanf("%d",&v[i]);

}

soma\_maior\_menor(v,&maior,&menor,&soma);

printf("maior : %d, menor: %d soma: %d",maior,menor,soma);

return 0;

}

146.Escreva um programa em linguagem C que solicita ao usuário a quantidade de alunos de uma turma e aloca um vetor de notas (números reais). Depois de ler as notas, imprime a média aritmética. Não deve haver desperdício de memória; e após ser utilizada, a memória deve ser devolvida.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int i=0,n;

double \* notas;

double media\_aritmetica;

printf("quantidade de alunos na turma: ");

scanf("%d",&n);

/\* alocando memória \*/

notas=(int \*)malloc(n\*sizeof(int));

for(i=0;i<n;i++){

printf("\nnotas[%d]:",i);

scanf("%lf",&notas[i]);

}

/\*calculando media aritmetica\*/

for(i=0;i<n;i++){

media\_aritmetica=media\_aritmetica+notas[i];

}

media\_aritmetica=media\_aritmetica/n;

printf("\nmedia aritmetica das notas: %.2f",media\_aritmetica);

free(notas);

return 0;

}

147.Crie um programa que:

147.1.Aloque dinamicamente um vetor

de cinco números inteiros.

147.2.Peça para o usuário digitar

os cinco números no espação alocado.

147.3.Mostre na tela os cinco

números.

147.4.Libere a memória alocada.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int \* v;

int i;

v=(int \*)malloc(5\*sizeof(int));

printf("Digite os 5 elementos:");

for (i=0;i<5;i++){

printf("\nv[%d]: ",i);

scanf("%d",v+i);

}

printf("Impressão de elementos:");

for(i=0;i<5;i++){

printf("\nv[%d]: %d",i,v[i]);

}

free(v);

return 0;

}

148.Desenvolva um programa que calcule a soma de duas matrizes MxN de número

reais (double). A implementação deste Programa deve considerar as dimensões

fornecidas pelo usuário. Dica: represente a matriz através de variáveis do tipo

double\*\*, usando alocação dinâmica de memória.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int m,n,i,j;

double \*\* matriz1;

double \*\* matriz2;

double \*\* matrizSoma;

printf("(m)linhas:");

scanf("%d",&m);

printf("(n)colunas:");

scanf("%d",&n);

/\*montagem das matrizes \*/

matriz1=(double\*\*)malloc(m\*sizeof(double\*));

matriz2=(double\*\*)malloc(m\*sizeof(double\*));

matrizSoma=(double\*\*)malloc(m\*sizeof(double\*));

for(i=0;i<m;i++){

matriz1[i]=(double \*)malloc(n\*sizeof(double));

matriz2[i]=(double \*)malloc(n\*sizeof(double));

matrizSoma[i]=(double \*)malloc(n\*sizeof(double));

}

/\*Preencher as matrizes 1 e 2 \*/

for(i=0;i<m;i++){

for(j=0;j<n;j++){

printf("\nmatriz1[%d][%d]:",i,j);

scanf("%lf",&matriz1[i][j]);

}

}

for(i=0;i<m;i++){

for(j=0;j<n;j++){

printf("\nmatriz2[%d][%d]:",i,j);

scanf("%lf",&matriz2[i][j]);

}

}

/\*A matriz soma é a soma de cada componente correspondente

da matriz 1 e 2\*/

for(i=0;i<m;i++){

for(j=0;j<n;j++){

matrizSoma[i][j]=matriz1[i][j]+matriz2[i][j];

}

}

/\*mostrar na tela a matriz soma \*/

for(i=0;i<m;i++){

for(j=0;j<n;j++){

printf("\nmatrizSoma[%d][%d]: %f",i,j,matrizSoma[i][j]);

}

}

/\* liberar memória das matrizes\*/

for(i=0;i<m;i++){

free(matriz1[i]);

free(matriz2[i]);

free(matrizSoma[i]);

}

free(matriz1);

free(matriz2);

free(matrizSoma);

return 0;

}

149.Faça um programa que leia do usuário o tamanho de um vetor a ser lido e faça a

alocação dinâmica de memória. Em seguida, leia do usuário seus valores e imprima

o vetor lido.

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

int i,n;

int \* v;

printf("tamanho do vetor:");

scanf("%d",&n);

/\*alocando memória\*/

v=(int\*)malloc(n\*sizeof(int));

for(i=0;i<n;i++){

printf("\nv[%d]: ",i);

scanf("%d",v+i);

}

/\*mostrar os valores\*/

for(i=0;i<n;i++){

printf("\nv[%d]: %d",i,v[i]);

}

free(v);

return 0;

}

150.Faça um programa que leia do usuário o tamanho de um vetor a ser lido e faça a

alocação dinâmica de memória. Em seguida, leia do usuário seus valores e mostre

quantos dos números são pares e quantos são ímpares.

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

int i,n;

int \* v;

int contPar=0,contImpar=0;

printf("tamanho do vetor:");

scanf("%d",&n);

/\*alocando memória\*/

v=(int\*)malloc(n\*sizeof(int));

for(i=0;i<n;i++){

printf("\nv[%d]: ",i);

scanf("%d",v+i);

}

/\*verificar se é ímpar ou par\*/

for(i=0;i<n;i++){

if(v[i]%2==0){

contPar++;

}

else{

contImpar++;

}

}

printf("pares: %d ímpares: %d",contPar,contImpar);

free(v);

return 0;

}

151.Faça um programa que receba do usuário o tamanho de uma string e chame uma

função para alocar dinamicamente essa string. Em seguida, o usuário deverá

informar o conteúdo dessa string. O programa imprime a string sem suas vogais.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int n,k=0,i;

char \* palavra;

char \* palavra\_sem\_vogais;

printf("Digite o tamanho: ");

scanf("%d\n",&n);

/\*alocar memoria\*/

palavra=(char \*)malloc((n+1)\*sizeof(char));

palavra\_sem\_vogais=(char \*)malloc((n+1)\*sizeof(char));

/\* receber do usuário a string\*/

scanf("%[^\n]",palavra);

/\*buscar vogais e nao incluir na nova cadeia de carateres\*/

for(i=0;i<n;i++){

if((palavra[i]=='a')||(palavra[i]=='e')||(palavra[i]=='i')||(palavra[i]=='o')||(palavra[i]=='u')){

continue;

}

else{

palavra\_sem\_vogais[k]=palavra[i];

k++;

}

}

/\*mostrar a nova cadeia de carateres \*/

if(k==0){

printf("todas são vogais! %s",palavra);

}

else{

printf("%s",palavra\_sem\_vogais);

}

return 0;

}

152.Faça um programa que leia um número N e:

• Crie dinamicamente e leia um vetor de inteiro de N posições;

• Leia um número inteiro X e conte e mostre os múltiplos desse número que

existem no vetor.

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

int N,X,contMult=0,i,k=0;

int \* nums;

int \* multX;

printf("(N)posicoes:");

scanf("%d",&N);

printf("valor para encontrar multiplos(X):");

scanf("%d",&X);

nums=(int \*)malloc(N\*sizeof(int));

multX=(int \*)malloc(N\*sizeof(int));

for(i=0;i<N;i++){

printf("nums[%d]:",i);

scanf("%d",nums+i);

}

/\* agora vamos verificar os multiplos de X\*/

for(i=0;i<N;i++){

if(nums[i]>=X){

if(nums[i]%X==0){

multX[k]=nums[i];

k++;

}

else{

continue;

}

}

else{

if(X%nums[i]==0){

multX[k]=nums[i];

k++;

}

else{

continue;

}

}

}

if(k==0){

printf("não existem multiplos de %d neste conjunto!!",X);

}

else{

printf("a quantidade de multiplos é %d e são :",k);

for(i=0;i<k;i++){

printf("\nmult[%d]: %d",i,multX[i]);

}

}

return 0;

}

153.Faça um programa que simule a memória de um computador: o usuário irá

especificar o tamanho da memoria, ou seja, quantos bytes serão alocados do tipo

inteiro. Para tanto, a memoria solicitada deve ser um valor múltiplo do tamanho do

tipo inteiro. Em seguida, o usuário terá duas opções: inserir um valor em uma

determinada posição ou consultar o valor contido em uma determinada posição. A

memoria deve iniciar com todos os dados zerados.

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

int tam\_memoria=0,posV,i,resp=1;

int \* v;

/\* pedir do usuario o tamanho necessario \*/

printf("1 inteiro ocupa 4 bytes: ");

scanf("%d",&tam\_memoria);

posV=tam\_memoria/4; /\* assim posso saber que se o tamanho e 12 bytes ele pode armazenar 3 inteiros \*/

/\* alocar o espaço necessario e inicializar com 0 \*/

v=(int \*)malloc(tam\_memoria);

for(i=0;i<posV;i++){

v[i]=0;

}

while(resp!=0){

printf("\n0- sair // 1- inserir valor em posição i // 2- ver o valor em um posição i");

scanf("%d",&resp);

if(resp==1){

printf("pos inserir (i):");

scanf("%d",&i);

printf("v[%d]:",i);

scanf("%d",&v[i]);

}

else if(resp==2){

printf("pos ver (i):");

scanf("%d",&i);

printf("v[%d]:",i);

scanf("%d",&v[i]);

}

else{

break;

}

}

return 0;

}

154.Escreva um programa que leia primeiro os 6 números gerados pela loteria e depois

os 6 números do seu bilhete. O programa então compara quantos números o

jogador acertou. Em seguida, ele aloca espaço para um vetor de tamanho igual a

quantidade de números corretos e guarda os números corretos nesse vetor.

Finalmente, o programa exibe os números sorteados e os seus números corretos.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int i;

int loteria[6];

int bilhete[6];

int \* acertos;

int k=0,contAcertos=0;

/\* ler os valores da loteria \*/

for(i=0;i<6;i++){

printf("loteria[%d]:",i);

scanf("%d",&loteria[i]);

}

/\* ler os valores do bilhete \*/

for(i=0;i<6;i++){

printf("bilhete[%d]:",i);

scanf("%d",&bilhete[i]);

}

/\* comparar os valores de loteria e bilhete termo a termo \*/

for(i=0;i<6;i++){

if(loteria[i]==bilhete[i]){

contAcertos++;

}

}

if(contAcertos==0){

printf("Não houve acertos ! o vetor de acertos não pode ser formado.");

}

else{

acertos=(int \*)malloc(contAcertos\*sizeof(int));

for(i=0;i<6;i++){

if(loteria[i]==bilhete[i]){

acertos[k]=loteria[i];

k++;

}

}

/\* mostrar na tela o resultado dos acertos e da loteria \*/

for(i=0;i<6;i++){

printf("\nloteria[%d]: %d",i,loteria[i]);

}

for(i=0;i<k;i++){

printf("\nresultado do bilhete [%d]: %d",i,acertos[i]);

}

}

return 0;

}

155.Faça um programa para armazenar em memória um vetor de dados contendo

1500 valores do tipo int, usando a função de alocação dinâmica de memória

CALLOC:

155.1.Faça um loop e verifique se o vetor contém realmente os 1500 valores

iniciados com zero (conte os 1500 zeros do vetor)

155.2.Atribua para cada elemento do vetor o valor do seu índice junto a este vetor.

155.3.Exibir na tela os 10 primeiros e os 10 últimos elementos do vetor.

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int \* v;

int i=0;

/\* alocando memoria e inicializando com zeros \*/

v=(int \*)calloc(1500,4);

for(i=0;i<1500;i++){

if(v[i]!=0){

printf("deu algo errado!!!!!");

}

}

for(i=0;i<1500;i++){

v[i]=i;

}

for(i=0;i<10;i++){

printf("\nv[%d]:%d",i,v[i]);

}

for(i=1499;i>1490;i--){

printf("\nv[%d]:%d",i,v[i]);

}

return 0;

}