LISTA DE EXERCÍCO DE PROGRAMAÇÃO

PROFESSOR: AGOSTINHO BRITO

Questão 1:

#include <stdio.h>

int main() {

int i=3,j=5;

int \*p, \*q;

p = &i;

q = &j;

printf("%d \n", p == &i); // resultado igual a 1//

printf("%d \n", \*p - \*q); // resultado igual a -2//

printf("%d \n", \*\*&p); // resultado igual a 3//

printf("%d \n", 3 - \*p / (\*q) + 7); // resultado igual a 10//

}

Questão 2:

#include <stdio.h>

int main() {

int i=5, \*p;

p = &i;

printf("%x \n", p); // posição na memoria 4094//

printf("%d \n", \*p+2); // \*p = i, i=5, 5+2 = 7 //

printf("%d \n", \*\*&p); // \*p = i, i = 5//

printf("%d \n", 3\*\*p); // 3 x \*p ( \*p = i, i = 5)= 15//

printf("%d \n", \*\*&p+4); // \*p (\*p = i=5)+ 4 = 9//

}

Questão 3:

p = i; - ilegal

q = &j; - legal

p = &\*&i; - legal

i = (\*&)j; - legal

i = \*&j; C - legal

i = \*&\*&j; - legal

q = \*p; - ilegal

i = (\*p)++ + \*q; - legal

Questão 4:

#include <stdio.h>

int main() {

int valor;

int \*p1;

float temp;

float \*p2;

char aux;

char \*nome = "Ponteiros";

char \*p3;

int idade;

int vetor[3];

int \*p4;

int \*p5;

/\* (a) \*/

valor = 10; // indica que valor = 10//

p1 = &valor; //guarda o endereço no ponteiro ( p1 mostra "valor")//

\*p1 = 20; //altera o armazemento do ponteiro p1 para 20//

printf("%d \n", valor); //motra o valor salvo no ponteiro//

/\* (b) \*/

temp = 26.5; // indica que valor = 26.5//

p2 = &temp; //guarda o endereço no ponteiro ( p2 mostra "temp")//

\*p2 = 29.0;//altera o armazemento do ponteiro p1 para 20//

printf("%.1f \n", temp);//motra o valor salvo no ponteiro ( o .1f ( uma casa decimal do tipo float//

/\* (c) \*/

p3 = &nome[0]; // guarda o vetor nome[0] ( primeiro elemento)no ponteiro 3//

aux = \*p3; //variável AUX aponta pro conteudo de p3//

printf("%c \n", aux); // imprime o valor guardado em AUX//

/\* (d) \*/

p3 = &nome[4]; //guarda o vetor nome[0] ( quarto elemento)no ponteiro 3//

aux = \*p3; //variável AUX aponta pro conteudo de p3//

printf("%c \n", aux); //imprime o valor guardado em AUX//

/\* (e) \*/

p3 = nome; // guarda o vetor nome (primeiro elemento)no ponteiro 3//

printf("%c \n", \*p3);// imprime o valor guardado em p3//

/\* (f) \*/

p3 = p3 + 4; // o endereço modificará 4 casa a frente do ponteito P3//.

printf("%c \n", \*p3); // imprime o valor mostrado por p3//

/\* (g) \*/

p3--; // o endereço modificará 4 casa para traz do ponteiro p3//

printf("%c \n", \*p3);// imprime o valor mostrado por p3//

/\* (h) \*/

vetor[0] = 31; // armaz. '31'

vetor[1] = 45;

vetor[2] = 27;

p4 = vetor;

idade = \*p4;

printf("%d \n", idade);

/\* (i) \*/

p5 = p4 +1;

idade = \*p5;

printf("%d \n", idade);

/\* (j) \*/

p4 = p5 + 2;

idade = \*p4;

printf("%d \n", idade);

/\* (l) \*/

p4 = p4 - 2;

idade = \*p4;

printf("%d \n", idade);

/\* (m) \*/

p5 = &vetor[2] - 1;

printf("%d \n", \*p5);

/\* (n) \*/

p5++;

printf("%d \n", \*p5);

return(0);

}

Questão 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| contador/valor/valor/endereco/endereo | | | | |
|  | | | i = 0vet[0] = 1.1\*(f + 0) = 1.1&vet[0] = 61FEB4(f + 0) = 61FEB4 | | |
|  | | i = 1vet[1] = 2.2\*(f + 1) = 2.2&vet[1] = 61FEB8(f + 1) = 61FEB8 | | | |
|  | i = 2vet[2] = 3.3\*(f + 2) = 3.3&vet[2] = 61FEBC(f + 2) = 61FEBC | | | | |
|  | | | | i = 3vet[3] = 4.4\*(f + 3) = 4.4&vet[3] = 61FEC0(f + 3) = 61FEC0 | |
|  | i = 4vet[4] = 5.5\*(f + 4) = 5.5&vet[4] = 61FEC4(f + 4) = 61FEC4  Questão 6:  \*(Pulo +2)  Questão 7  p = mat + 1; - certa  p = mat++; - errada  p = ++mat; - errada  x = (\*mat)++; - errada  Questão 8:  int main(){  int vet[] = {4,9,13, 15, 16}; // declara o vetor e seus valores//  int i; // declara que i é do tipo int//  for(i=0;i<4;i++) //  {  printf("%d ",\*(vet+i)); // imprime 3 elementos do vetor//  }  }  B)  #include <stdio.h>  int main(){  int vet[] = {4,9,13};  int i;  for(i=0;i<3;i++){  printf("%X ",vet+i); // imprime o endereço do vetor//  }  }  Questão 9  x para char - x + 1 = 4093, x + 2 = 4094, x + 3 = 4095  x para int - x + 1 = 4094, x + 2 = 4096, x + 3 = 4098  x para float - x + 1 = 4096, x + 2 = 409A, x + 3 = 409E  x para double - x + 1 = 409A, x + 2 = 40A2, x + 3 = 40AA  Questão 10  Questão 11 | | | | |
|  | | | | |  |

aloha[2] = value – Apresenta falha

scanf("%f", &aloha); - sem falha

aloha = value; - apresenta falha

printf("%f", aloha); - sem falha

coisas[4][4] = aloha[3]; - sem falha

coisas[5] = aloha; - apresenta falha

pf = value; - apresenta falha

pf = aloha; - sem falha

**Questão 12**

**Questão 13**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n,i,j;

float \*a, t;

printf("quantos numeros voce quer ordenar: ");

scanf("%i",&n);

a=(float \*)malloc(n \*sizeof(float));

printf("\nDigite %i numeros: \n\n",n);

for(i=0;i<=n-1;i++)

{

scanf("%f", (a+i));

}

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

{

for(j=0;j<=i;j++)

{

if(\*(a+i)<\*(a+j))

{

t=\*(a+i);

\*(a+i)=\*(a+j);

\*(a+j)=t;

}

}

}

printf("\nDepois de ordenar em ordem crescente: \n");

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

printf("\n%f",\*(a+i));

free(a);

return 0;

}

**Questão 14**

#include<stdio.h>

#include<stdlib.h>

int cmpfunc (const void \* a, const void \* b) {

return ( \*(float\*)a - \*(float\*)b );

}

int main()

{

int n,i;

float \*a;

printf("quantos numeros voce quer ordenar: ");

scanf("%i",&n);

a=(float \*)malloc(n \*sizeof(float));

printf("\nDigite %i numeros: \n\n",n);

for(i=0;i<=n-1;i++)

{

scanf("%f", (a+i));

}

qsort(a, n, sizeof(float), cmpfunc);

printf("\nDepois de ordenar em ordem crescente: \n");

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

printf("\n%f",\*(a+i));

free(a);

return 0;

}

**Questão 15**

#include<stdio.h>

#include<stdlib.h>

int cmpfunc (const void \* a, const void \* b) {

return ( \*(float\*)a - \*(float\*)b );

}

void minhafunccomp(float \*base, size\_t nitems, float (\*compar)(const void \*, const void\*)){

int i, j;

float t;

for(i=0;i<nitems;i++)

{

for(j=0;j<=i;j++)

{

if(\*(base+i)<\*(base+j))

{

t=\*(base+i);

\*(base+i)=\*(base+j);

\*(base+j)=t;

}

}

}

}

int main()

{

int n,i;

float \*a;

printf("quantos numeros voce quer ordenar: ");

scanf("%i",&n);

a=(float \*)malloc(n \*sizeof(float));

printf("\nDigite %i numeros: \n\n",n);

for(i=0;i<=n-1;i++)

{

scanf("%f", (a+i));

}

minhafunccomp(a, n, cmpfunc);

printf("\nDepois de ordenar em ordem crescente: \n");

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

printf("\n%f",\*(a+i));

free(a);

return 0;

}

**Questão 16**

#include<stdio.h>

#include<stdlib.h>

#include <time.h>

int cmpfunc (const void \* a, const void \* b) {

return ( \*(float\*)a - \*(float\*)b );

}

int main()

{

clock\_t tic = clock();

int n,i;

float \*a;

float valores[] = { 88, 56, 100, 2, 25, 65 ,642, 656, 65689, 16, 33, 18 };

n = 12;

a=(float \*)malloc(n \*sizeof(float));

a = &valores;

qsort(a, n, sizeof(float), cmpfunc);

printf("\nDepois de ordenar em ordem crescente: \n");

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

printf("\n%f",\*(a+i));

clock\_t toc = clock();

printf("\nTempo de execucao: %f segundos\n", (double)(toc - tic) / CLOCKS\_PER\_SEC);

return 0;

}

**B**

#include<stdio.h>

#include<stdlib.h>

#include <time.h>

int cmpfunc (const void \* a, const void \* b) {

return ( \*(float\*)a - \*(float\*)b );

}

void minhafunccomp(float \*base, size\_t nitems, float (\*compar)(const void \*, const void\*)){

int i, j;

float t;

for(i=0;i<nitems;i++)

{

for(j=0;j<=i;j++)

{

if(\*(base+i)<\*(base+j))

{

t=\*(base+i);

\*(base+i)=\*(base+j);

\*(base+j)=t;

}

}

}

}

int main()

{

clock\_t tic = clock();

int n,i;

float \*a;

float valores[] = { 88, 56, 100, 2, 25, 65 ,642, 656, 65689, 16, 33, 18 };

n = 12;

a=(float \*)malloc(n \*sizeof(float));

printf("\nDigite %i numeros: \n\n",n);

a = &valores;

minhafunccomp(a, n, cmpfunc);

printf("\nDepois de ordenar em ordem crescente: \n");

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

printf("\n%f",\*(a+i));

clock\_t toc = clock();

printf("\nTempo de execucao: %f segundos\n", (double)(toc - tic) / CLOCKS\_PER\_SEC);

return 0;

}

O exemplo A foi bem mais rápido do que o exemplo B.

**Questão 17**

#include<stdio.h>

#include<stdlib.h>

int main(){

int n,i,\*vetor1, \*vetor2, \*vetor3;

printf("Digite a quantidade de elementos em cada vetor: ");

scanf("%i",&n);

vetor1=(int\*)malloc(n\*sizeof(int));

vetor2=(int\*)malloc(n\*sizeof(int));

vetor3=(int\*)malloc(n\*sizeof(int));

printf("Digite os elementos do vetor 1: ");

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

{

scanf("%i",vetor1+i);

}

printf("Digite os elementos do vetor 2: ");

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

{

scanf("%i",vetor2+i);

}

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

\*(vetor3+i) = \*(vetor1+i) + \*(vetor2+i);

}

printf("\nVetor 3: \n");

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

printf("%i ", \*(vetor3+i));

}

free(vetor1);

free(vetor2);

free(vetor3);

return 0;

}

**Questão 18**

#include <stdio.h>

#include <stdlib.h>

// Here, the parameter is an array of pointers

void matMult(int\*\* matA, int\*\* matB, int\*\* matC, int nLinA, int nColA, int nColB)

{

int i,j,k;

for(i=0;i<nLinA;i++)

{

for(j=0;j<nColB;j++)

{

matC[i][j]=0;

for(k=0;k<nColA;k++)

{

matC[i][j]+=matA[i][k]\*matB[k][j];

}

}

}

}

// Program to pass the 2D array to a function in C

int main(void)

{

int nLinA, nColA, nColB;

printf("Digite o numero de linhas da matriz A=");

scanf("%d",&nLinA);

printf("Digite o numero de colunas da matriz A=");

scanf("%d",&nColA);

printf("Digite o numero de colunas da matriz B=");

scanf("%d",&nColB);

// dynamically create an array of pointers of size `m`

int \*\*matA = (int \*\*)malloc(nLinA \* sizeof(int \*));

// dynamically allocate memory of size `n` for each row

for (int r = 0; r < nLinA; r++) {

matA[r] = (int \*)malloc(nColA \* sizeof(int));

}

// dynamically create an array of pointers of size `m`

int \*\*matB = (int \*\*)malloc(nColA \* sizeof(int \*));

// dynamically allocate memory of size `n` for each row

for (int r = 0; r < nColA; r++) {

matB[r] = (int \*)malloc(nColB \* sizeof(int));

}

// dynamically create an array of pointers of size `m`

int \*\*matC = (int \*\*)malloc(nLinA \* sizeof(int \*));

// dynamically allocate memory of size `n` for each row

for (int r = 0; r < nLinA; r++) {

matC[r] = (int \*)malloc(nColB \* sizeof(int));

}

printf("enter the first matrix element=\n");

for(int i=0;i<nLinA;i++)

{

for(int j=0;j<nColA;j++)

{

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

}

}

printf("enter the first matrix element=\n");

for(int i=0;i<nColA;i++)

{

for(int j=0;j<nColB;j++)

{

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

}

}

matMult(matA, matB, matC, nLinA, nColA, nColB);

// print 2D array

for (int i = 0; i < nLinA; i++)

{

for (int j = 0; j < nColB; j++) {

printf("%3d", matC[i][j]);

}

printf("\n");

}

//deallocate memory

for (int i = 0; i < nLinA; i++) {

free(matA[i]);

}

free(matA);

for (int i = 0; i < nColA; i++) {

free(matB[i]);

}

free(matB);

for (int i = 0; i < nLinA; i++) {

free(matC[i]);

}

free(matC);

return 0;

}