**Organização dos Dados na Memória**

**EXERCÍCIOS**

Defina a saída de cada trecho de código. Utilize endereços fictícios da planilha que simula a memória para resolver os exercícios.

Sugestão: usar template 03 da planilha.

**Ex 01**

//[endereçamento de memória]

int i = 10; //sugestão de endereço: 0

char c = 'a'; //sugestão de endereço: 10

float f = 2.5; //sugestão de endereço: 20

printf("i: %d \n", i); = 10

printf("&i: %u \n", &i); = 0

printf("c: %c \n", c); = a

printf("&c: %u' \n", &c); = 10

printf("f: %f \n", f); = 2.5

printf("&f: %u \n", &f); = 20

**Ex 02**

//[endereçamento de arrays unidimensionais][aritmética de ponteiro]

int a = 100; //sugestão de endereço: 32

int v[4] = {10,20,30,40}; //sugestão de endereço: 40

printf("a: %d \n", a); = 100

printf("&a: %u \n", &a); = 32

printf("v: %d \n", v); = 40

printf("&v: %u \n", &v); = 40

printf("&v[0]: %u \n", &v[0]); = 40

printf("&v[1]: %u \n", &v[1]); = 44

printf("&v[2]: %u \n", &v[2]); = 48

printf("&v[3]: %u \n", &v[3]); = 52

printf("v+0: %u \n", v+0); = 40

printf("v+1: %u \n", v+1); = 44

printf("v+2: %u \n", v+2); = 48

printf("v+3: %u \n", v+3); = 52

**Ex 03**

//[endereçamento de strings][aritmética de ponteiro]

char nome[10] = "Joao"; //sugestão de endereço: 64

char fruta[] = "laranja"; //sugestão de endereço: 78

printf("sizeof(nome): %ld \n", sizeof(nome)); (retorna em bytes) = 10

printf("sizeof(fruta): %ld \n", sizeof(fruta)); = 8

printf("fruta: %u \n", fruta); = laranja

printf("&fruta: %u \n", &fruta); = 78

printf("&fruta[0]: %u \n", &fruta[0]); = 78

printf("&fruta[1]: %u \n", &fruta[1]); = 79

printf("&fruta[2]: %u \n", &fruta[2]); = 80

printf("fruta+0: %u \n", fruta+0); = 78 (pula o vetor de byte em byte)

printf("fruta+1: %u \n", fruta+1); = 79

printf("fruta+2: %u \n", fruta+2); = 80

printf("fruta+0: %u \n", &fruta+0); = 78 (percorre toda string até o final)

printf("fruta+1: %u \n", &fruta+1); = 86

printf("fruta+2: %u \n", &fruta+2); = 92

**Ex 04**

//[endereçamento de arrays multidimensionais][aritmética de ponteiro]

short int m[3][3] = {{10,11,12}, {20,21,22}, {31,32,33}}; //sugestão de endereço: 96

printf("m: %u \n", m); = 96

printf("&m: %u \n", &m); = 96

printf("&m[0][0]: %u \n", &m[0][0]); = 96

printf("&m[0][1]: %u \n", &m[0][1]); = 98

printf("&m[1][0]: %u \n", &m[1][0]); = 102

printf("&m[1][1]: %u \n", &m[1][1]); = 104

printf("&m[2][0]: %u \n", &m[2][0]); = 108

printf("&m[2][1]: %u \n", &m[2][1]); = 110

printf("m+0: %u \n", m+0); = 96

printf("m+1: %u \n", m+1); = 102

printf("m+2: %u \n", m+2); = 108

printf("m[0]: %u \n", m[0]); = 96

printf("m[1]: %u \n", m[1]); = 102

printf("m[2]: %u \n", m[2]); = 108

printf("m[1]+0: %u \n", m[1]+0); = 102

printf("m[1]+1: %u \n", m[1]+1); = 104

printf("m[1]+2: %u \n", m[1]+2); = 106

**Ex 05**

//[endereçamento de structs]

struct aluno

{

short int ra;

char nome[10];

float nota

};

struct aluno a; //sugestão de endereço: 128

printf("&a: %u \n", &a); = 128

printf("&a.ra: %u \n", &a.ra); = 128

printf("&a.nome: %u \n", &a.nome); = 130

printf("&a.nota: %u \n", &a.nota); = 140

**Ex 06**

//[indireção]

int x = 20; //sugestão de endereço: 160

int \*p1 = &x; //sugestão de endereço: 168

int \*\*p2 = &p1; //sugestão de endereço: 176 também são 8 bytes?

int \*\*\*p3 = &p2; //sugestão de endereço: 184

printf("x: %d \n", x); = 20

printf("&x: %d \n", &x); = 160

printf("p1: %d \n", p1); = 160

printf("&p1: %d \n", &p1); = 168

printf("\*p1: %d \n", \*p1); = 20

printf("p2: %d \n", p2); = 168

printf("&p2: %d \n", &p2); = 176

printf("\*p2: %d \n", \*p2); = 160

printf("\*\*p2: %d \n", \*\*p2); = 20

printf("p3: %d \n", p3); = 176 por que não 168?

printf("&p3: %d \n", &p3); = 184

printf("\*p3: %d \n", \*p3); = 168

printf("\*\*p3: %d \n", \*\*p3); = 160

printf("\*\*\*p3: %d \n", \*\*\*p3); = 20

**Ex 07**

//[ponteiro vs vetor][aritmética de ponteiro]

short int i = 1; //sugestão de endereço: 192

short int v[4]; //sugestão de endereço: 200

while (i <= 4){

\*(v + i) = i++;

}

**Ex 08**

//[endereçamento de struct]

struct produto{

int codigo;

char descricao[8];

float preco;

};

struct produto p1; //sugestão de endereço: 224

struct produto p2; //sugestão de endereço: 240

p1.codigo = 1;

strcpy(p1.descricao, "Mouse");

p1.preco = 25.50;

p2 = p1;

p2.codigo = 2;

**Ex 09**

//[ponteiro de struct]

struct aluno{

short int ra;

short int notas[2];

};

typedef struct aluno Aluno;

Aluno a1 = {1, {85,90}}; //sugestão de endereço: 256

Aluno \*p1 = &a1; //sugestão de endereço: 266

short int \*v = (\*p1).notas;

v[1] = 55;

printf("a1.ra: %d\n", a1.ra);

printf("a1.notas: %d, %d\n", a1.notas[0], a1.notas[1]);

**Ex 10**

//[atribuição de ponteiros] [aritmética de ponteiros]

short int v1[] = {1,2,3}; //sugestão de endereço: 288

short int v10[3] = {1,2,3}; //sugestão de endereço: 296

short int \*v2 = v1; //sugestão de endereço: 304

v2[0] = 10;

\*(v2+1) = 20; //\*(v2+1) avança para o proximo dado. V2+1 = acrescenta unidade ao dado.

short int i; //sugestão de endereço: 314

for(i=0; i<3; i++){

printf("%d ", v1[i]);

}

**Ex 11**

// [void\*] [casting]

struct produto{

int codigo;

char descricao[8];

float preco;

};

struct produto prod = {10,"memoria",100.55}; //sugestão de endereço: 320

void \*p; //sugestão de endereço: 336

p = &prod.codigo;

short int \*cod = (short int\*)p; //sugestão de endereço: 344

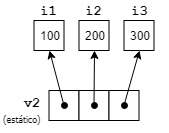
printf("Codigo: %d\n", \*cod);

p = &prod.preco;

printf("Preco: %.2f\n", \*(float\*)p);

**Ex 12**

Além de representar os dados na planilha de memória, escreva o trecho de código correspondente a estrutura ilustrada na imagem.



//sugestão de endereço:

// i1(short int): 352

// i2(short int): 354

// i3(short int): 382

// v2: 356

**Ex 13**

short int v[4] = {10,20,30,40}; //sugestão de endereço: 384

short int\* p = v + 1; //sugestão de endereço: 398

\*p \*=2;

p++;

\*(p+1) = 5;

**Ex 14**

struct aluno{

short int ra;

char nome[8];

short int notas[2];

};

typedef struct aluno Aluno;

Aluno a1 = {10,"Joao", {85, 90}}; //sugestão de endereço: 416

short int \*p1 = a1.notas + 1; //sugestão de endereço: 432

\*p1 = 100;

char\* p2 = a1.nome + 2; //sugestão de endereço: 440

\*p2 = '@';