

EXERCÍCIOS DE REVISÃO DE TÓPICOS DA LINGUAGEM C (ANSI)

1. Responda:

a) Você conhece as diretivas de compilação? Descreva as seguintes diretivas:

`#include`, `#define`, `#undef`, `#ifdef`, `#ifndef`, `#if`, `#else`, `#elif`, `#endif`

b) Leia sobre os comandos `memcpy` e `typedef`.

c) Para o código e representação de mapa de memória exibidos abaixo, responda:

Qual é o valor final de *p*?
Qual é o valor final de **p*?
Qual é o valor final de *x*?
Qual é o valor final de *&x*?
Qual é o valor final de *pp*?
Qual é o valor final de **pp*?
Qual é o valor final de ***pp*?
Qual é o valor final de *y*?

```
main(void)
{
    int *p=NULL, **pp=NULL, x = 321, y=101;
    p = &x;
    pp=&p;
    *p= -3;
    y=**pp;
}
```

Campo	Endereço	Conteúdo
p	00007FFC127C92C8	
x	00007FFC127C92C0	
y	00007FFC127C92C4	
pp	00007FFC127C92D0	
*pp == p		
**pp == *p == x		

d) Para o código e representação de mapa de memória exibidos abaixo, responda:

Qual é o valor de p ?

Qual é o valor de $p \rightarrow \text{self}$?

<pre>struct teste2{ int inteiro; float real; char nome[30]; struct teste2 *self; };</pre>	<pre>struct teste2 x = {115, 2.5, "Smith", NULL}, *p=NULL; p = &x; p->self = &x;</pre>
---	---

Campo	Endereço	Conteúdo
x	00007FFE0F4FB570	
p	00007FFE0F4FB568	
p->self	00007FFE0F4FB598	

e) Para o código e representação de mapa de memória exibidos abaixo, responda:

Qual é o valor de p ?

Qual é o valor de $\&(p \rightarrow \text{real})$?

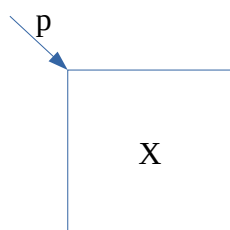
Qual é o valor de $p \rightarrow \text{real}$?

Qual é o valor de $\&(p \rightarrow \text{apont})$?

Qual é o valor de $p \rightarrow \text{apont}$?

Qual é o valor de $*(p \rightarrow \text{apont})$?

<pre>struct teste{ int inteiro; float real; char nome[30]; char rua[30]; int *apont; };</pre>	<pre>main(void) { struct teste *p=NULL, x = {321, 2.39, "Silva", "Timbo", NULL}; int y=101; p = &x; p->apont = &y; }</pre>
---	---



Campo	Endereço	Conteúdo
p	00007FFDE307C4A8	
p->inteiro	00007FFDE307C4B0	
p->real	00007FFDE307C4B4	
p->nome	00007FFDE307C4B8	
p->rua	00007FFDE307C4D6	
p->apont	00007FFDE307C4F8	
...	...	
y	00007FFDE307C4A4	

2. Para a *struct* descrita abaixo, encontre e solucione os erros/equívocos nos exemplos abaixo:

```
typedef struct teste{    int inteiro;
                        float real;
                        char nome[30];

                        } informacao;
```

- A)
- ```
main(void)
{
 informacao *p, x = {321, 2.39, "Silva"};
 if (p)
 printf("valores da struct X: %i, %f, %s", p->inteiro, p->real, p->nome);
 else
 printf("o ponteiro está anulado");
}
```
- B)
- ```
main(void)
{
    informacao *p, x = {321, 2.39, "Silva"};
    p= (struct teste *) malloc(sizeof(struct teste));
    printf("Campos da variável x: %i, %f, %s", p → inteiro, p → real, p->nome);

}
```
- C)
- ```
main(void)
{
 informacao *p, x = {321, 2.39, "Silva"};
 p = x;
 printf("Campos da variável x: %i, %f, %s", p → inteiro, p → real, p->nome);

}
```
- D)
- ```
main(void)
{
    informacao *p, x = {321, 2.39, "Silva"};
    p = &x;
    printf("Campos da variável x: %i, %f, %s", p.inteiro, p.real, p.nome);

}
```
- E)
- ```
main(void)
{
 informacao *p, x = {321, 2.39, "Silva"};
 p= (struct teste *) malloc(sizeof(struct teste));
 p=&x;
 printf("Campos da variável x: %i, %f, %s", p → inteiro, p → real, p->nome);

}
```
- F)
- ```
main(void)
{
    int a=10, vet[ ]={1,2,3,4,5}, *p=NULL;
    float b=35.75;
    informacao y = {31,"Wilson"};
    void *ptr; // Declaracao de um ponteiro para um tipo genérico (void)
    ptr=&a; // Atribuindo o endereço de um inteiro.
    printf("a = %d \n", * ( (int*) ptr ));
    ptr=&b; // Atribuindo o endereço de um float.
    printf("b = %f \n",*( (float*) ptr ));
    ptr= &y;
    printf("nome= %s, idade = %i \n\n", ((info*) ptr)->nome,((info*) ptr)->idade);
    printf("\n\n Acessando um vetor por aritmetica de ponteiro void\n");
    ptr=&vet[0];
    for (int i =0;i<6;i++,ptr++)
        printf("vet[%i] = %i \n", i, *( (int*) ptr ));
}
```

3. Escreva os comandos em linguagem C que levem: (i) da situação 'A' para a 'B' e (ii) da situação 'A' para a 'C', conforme está ilustrado na Figura 1, onde: *taminfo*, *dados*, *topo* e *abaixo* são campos de structs já criadas e representadas graficamente e o símbolo ' \rightarrow ' indica um local referenciado por um apontador.

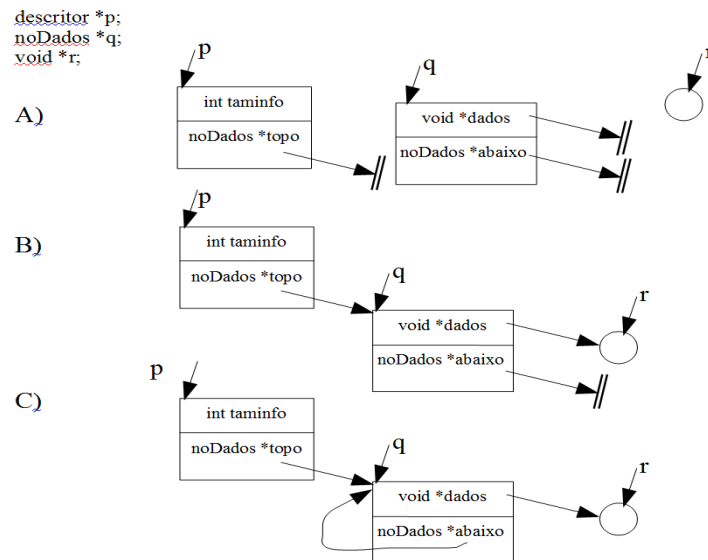


Figura 1: Configurações de nodos.

4. Na Figura 2 e na Figura 3, A e B são seqüências encadeadas sofrendo uma inserção. Nas figuras, *pt1*, *pt2* e *pt3* são apontadores já declarados para as respectivas "structs" contendo os campos de ligação: 'a' (aponta para o vizinho esquerdo) e/ou 'b' (aponta para o vizinho direito). Nestas condições (sem declarar ou alocar qualquer nova variável!) pede-se que você escreva o menor número de comandos, em linguagem C, para levar de A1 para A2 e de B1 para B2, conforme as figuras abaixo.

```
struct xxx { int w;  
            struct xxx *b;  
        }*pt3,*pt2,*pt1;
```

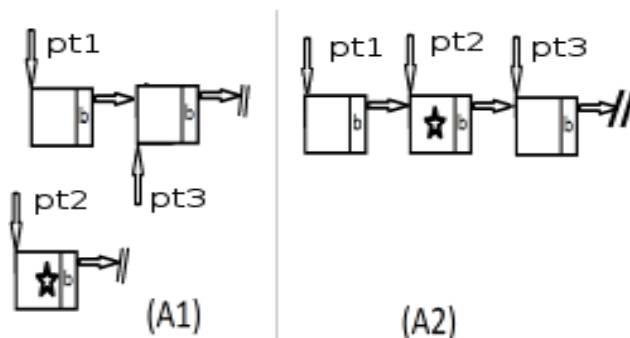


Figura 2: Seqüência simplesmente encadeada.

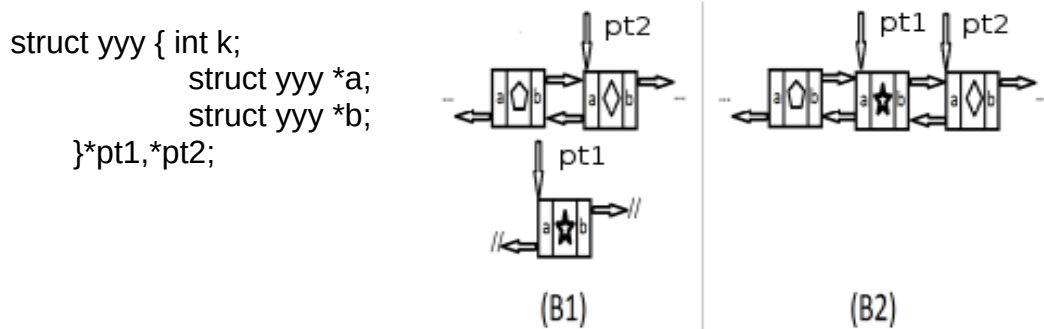


Figura 3: Sequência duplamente encadeada.

5. Escreva os comandos em linguagem C que, para cada caso, levem do estado 1 para o estado 2 na figura Figura 4:

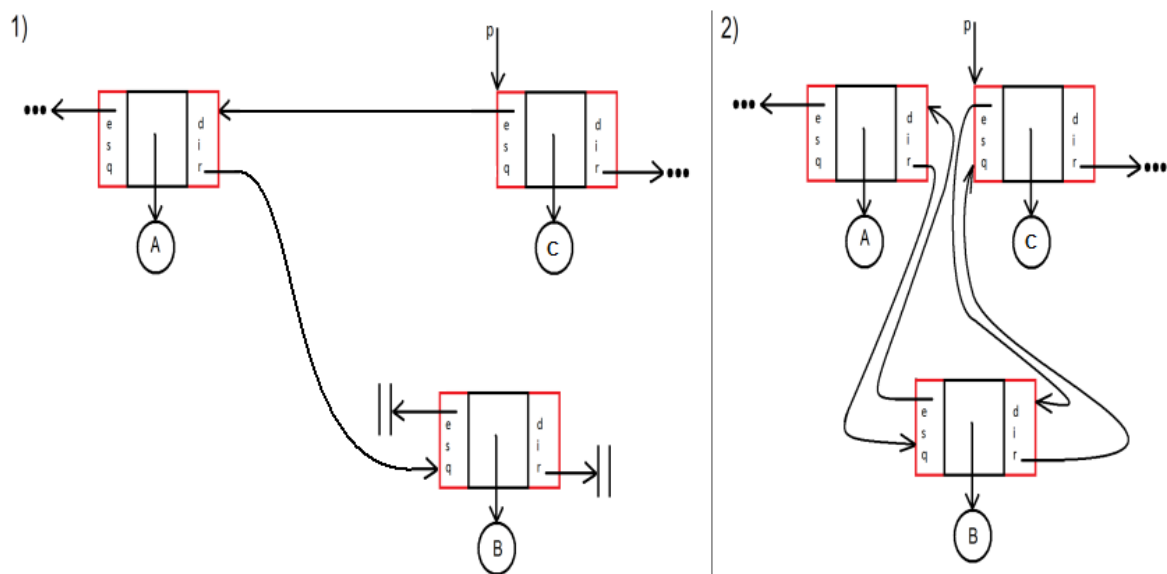


Figura 4: Inserção de elemento duplamente encadeado.

6. Construa a função `int contaNodo(struct nodo *p)` a qual retorna a contagem do número de nós atualmente inseridos em uma lista encadeada conforme a Figura 4 sequências encadeadas exibidas abaixo:

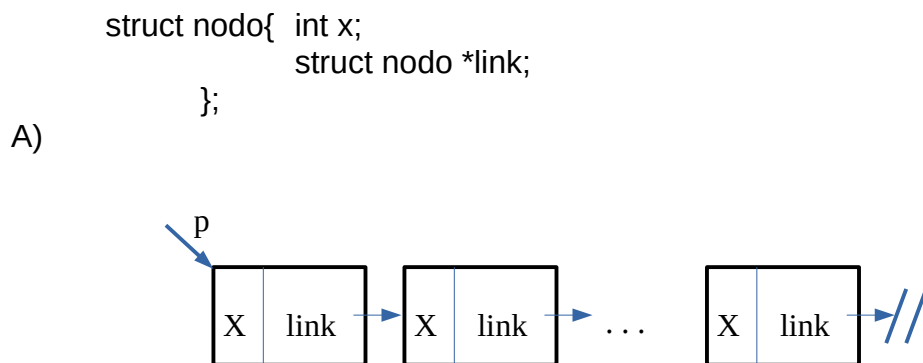


Figura 5: Sequência simplesmente encadeada, finalizada com o Null.

B)

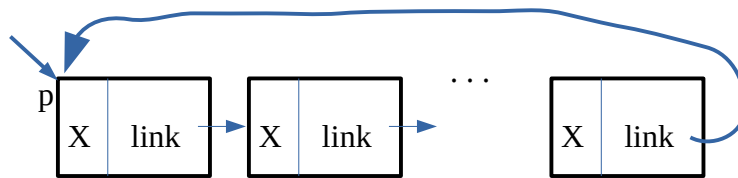


Figura 6: Sequência encadeada circular.

C)

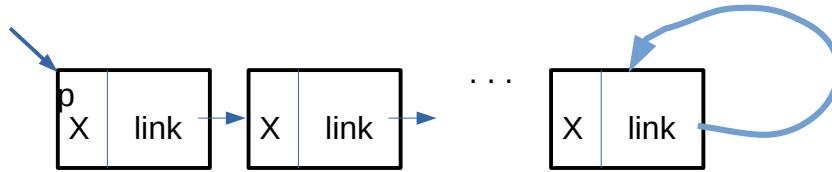


Figura 7: Sequência encadeada com finalização assinalada com um laço.

7. Um descritor é uma “struct” com metadados sobre a estrutura de dados propriamente dita (uma lista encadeada, por exemplo). Um exemplo de descritor pode ser visto na Figura 8 contendo os campos: *tamanho* (quantidade atual de elementos na lista) e *inicio* (apontador para o nó inicial da lista). Para uma lista simplesmente encadeada com descritor, construa as funções:

a) *int reinicia(descritor *p)*, a qual promove um “reset” da lista removendo todos os nós, retornando a lista ao estado “vazia” (*inicio == Null*). Sugestão: utilize ponteiro auxiliar como um “farejador” de final de lista dentro de um laço;

b) *int insere(descritor *p, Nodo *novo, int pos)*, a qual insere o novo item na posição *pos* em uma sequência simplesmente encadeada (exemplo na Figura 8). Um requisito adicional é que a posição *pos* já exista na sequência de nós de dados da lista.

As funções devem retornar zero ou um (1) a depender da respectiva operação falhar ou ter sucesso.

typedef struct nodo{struct data dt; struct nodo *link; }Nodo;	typedef struct descritor{ int tamanho; Nodo *inicio; }Descritor;
---	---

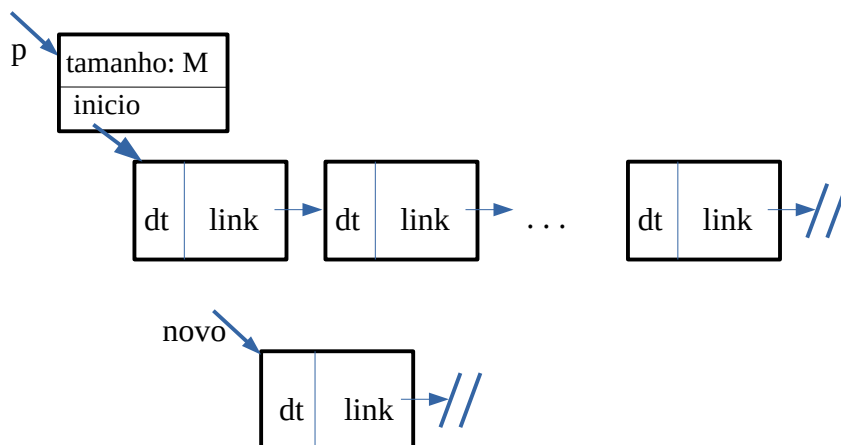
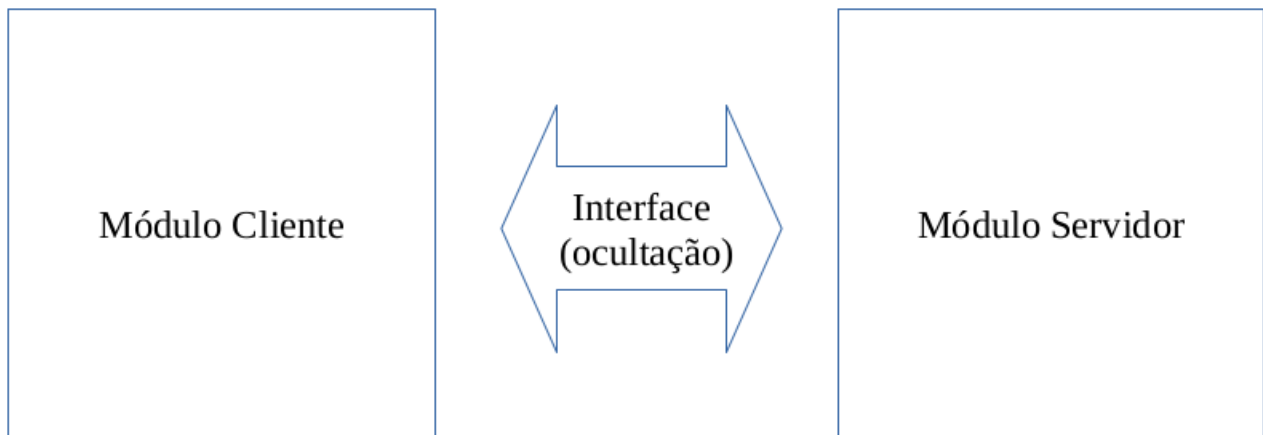


Figura 8: Sequência simplesmente encadeada com descritor.

8. Discuta a utilização da diretiva include para separar um sistema em arquivos diferentes visando a ocultação de implementação e dados (encapsulamento).



Respostas dos itens

1.c)

Campo	Endereço	Conteúdo
p	00007FFC127C92C8	00007FFC127C92C0
x	00007FFC127C92C0	-3
y	00007FFC127C92C4	-3
pp	00007FFC127C92D0	00007FFC127C92C8
*pp == p	&p	00007FFC127C92C0
**pp == *p == x	&x	-3

1.d)

Campo	Endereço	Conteúdo
&x	00007FFE0F4FB570	
p	00007FFE0F4FB568	00007FFE0F4FB570
p->self	00007FFE0F4FB598	00007FFE0F4FB570

1.e)

Campo	Endereço	Conteúdo
p	00007FFDE307C4A8	00007FFDE307C4B0
p->inteiro	00007FFDE307C4B0	321
p->real	00007FFDE307C4B4	2.39
p->nome	00007FFDE307C4B8	Silva
p->rua	00007FFDE307C4D6	Timbo
p->apont	00007FFDE307C4F8	00007FFDE307C4A4
...	...	
y	00007FFDE307C4A4	101