Técnicas de Desenvolvimento de Algoritmos
Prof. Walace Bonfim

Lista de Exercícios 2 – assuntos: ESTRUTURAS E PONTEIROS

LISTA PARA FIXAR O CONTEÚDO E ESTUDAR PARA A 2ª AVALIAÇÃO

NÃO PONTUA, MAS SERVE COMO RECURSO CASO VOCÊ PRECISE DE ATÉ 0,5 PONTO

Instruções para a resolução e entrega da lista.
1 – A entrega da lista é individual, entretanto incentivo o estudo em grupo, para compartilhamento de experiências e conhecimentos.
2 – As respostas devem ser dadas logo abaixo de cada questão, no próprio arquivo da lista.
3 – PRAZO – VEJA NA DESCRIÇÃO DA TAREFA NO BBOARD.

1. Quais são os valores mostrados pelo programa em C abaixo?

```c
int main(void){
    int i = 99, j;
    int *p;

    p = &i;
    j = *p + 100;
    printf("i = %d, j = %d, *p = %d", i, j,*p);
}
```

2. Quais são os valores mostrados pelo programa em C abaixo?

```c
int main() {
    int     a = 5, b = 12, c;
    int *   p;
    int *   q;

    p = &a;
    q = &b;

    printf("p = %x\tq = %x\n", p, q);

    c = *p + *q;

    printf("c = %d, a = %d, b = %d", c, a, b);
}
```

3. Quais são os valores mostrados pelo programa em C abaixo?

```c
int main() {
    int     a = 4, b = 3;
    int *   p1, * p2;

    p1 = &a;
    p2 = p1;

    printf("p1 = %x\tp2 = %x\n", p1, p2);

    *p2 = *p1 + 3;
    b = b * (*p1);
    (*p2)++;
    p1 = &b;

    printf("%d\t%d\n", *p1, *p2);
    printf("%d\t%d\n", a, b);
}
```

4. Quais são os valores mostrados pelo programa em C abaixo?

```c
void func(int * px, int * py){

    printf("*px = %d\t*py = %d\n", *px, *py);
    printf("px = %x\tpy = %x\n", px, py);
    printf("&px = %x\t&py = %x\n", &px, &py);

    px = py;
    *py = (*py) * (*px);
    *px = *px + 2;
}

int main() {
    int x, y;

    scanf("%d",&x); // entrar com o valor 3
    scanf("%d",&y); // entrar com o valor 4

    func(&x, &y);

    printf("x = %d, y = %d", x, y);
}
```

5. A memória do computador funciona, basicamente, armazenando sequências de bytes, e cada sequência tem seu próprio tipo e endereço de memória. A quantidade de bytes de uma sequência é determinada pelo tipo, que pode ser: tipo primitivo, ponteiro, estrutura de dados ou a combinação ilimitada destes. O endereço de memória é representado por um número hexadecimal na forma 0x1234 (os valores apresentados nesta questão são imaginários e não representam endereços reais válidos).

Com base nestas informações, avalie as afirmações a seguir.

I. Um ponteiro é um tipo de variável que pode ser manipulado com os operadores "&" e "*".
II. O que um ponteiro carrega dentro dele é um número hexadecimal, que é um endereço de memória de uma sequência de bytes de um determinado tipo.

III. É possível realizar operações aritméticas (soma, subtração, multiplicação e divisão) sobre ponteiros.
IV. Ao somarmos uma unidade a um ponteiro, este terá seu conteúdo somado da quantidade de bytes correspondente ao tipo da área para a qual ele aponta.

É correto o que se afirma em

A) I e IV, apenas.

B) III, apenas.

C) I e II, apenas.

D) II e III, apenas.

E) I, II e IV.

6. Observe o código abaixo e responda a pergunta.

```
1  void funcaoA (int ** pt_ptL, int * ptLista, int Lista) {
2      // faz qualquer coisa...
3  }
4
5  int main ()
6  {
7      int lst;
8
9      int * pl = (int *) malloc (sizeof(int));
10
11     lst = *pl;
12
13     funcaoA (&pl, pl, lst);
14 }
```

Considere que:

- na linha 9 o malloc retorna o endereço de memória 0x456;
- que a variável pl está alocada no endereço de memória 0x777;
- no main(), a função funcaoA está sendo chamada com passagem de parâmetros.

Técnicas de Desenvolvimento de Algoritmos
Prof. Walace Bonfim

Avalie as alternativas abaixo e escolha a única correta.

a)

no main()
- lst representa uma variável inteira
- pl carrega 0x456

na funcaoA()
- pt_ptL carrega 0x777
- ptLista carrega 0x456
- Lista é uma cópia de lst

b)

no main()
- lst representa uma estrutura
- pl carrega 0x777

na funcaoA()
- pt_ptL carrega 0x456
- ptLista carrega 0x456
- Lista é uma cópia de lst

c)

no main()
- lst carrega 0x456
- pl carrega 0x456

na funcaoA()
- pt_ptL carrega 0x777
- ptLista carrega 0x456
- Lista não é uma cópia de lst

d)

no main()
- lst é uma estrutura alocada em 0x456
- pl carrega 0x456

na funcaoA()
- pt_ptL carrega 0x777
- ptLista carrega 0x456
- Lista não é uma cópia de lst

e)

no main()
- lst é um ponteiro para a estrutura alocada em 0x456
- pl é a estrutura alocada em 0x456

na funcaoA()
- pt_ptL carrega 0x456
- ptLista carrega 0x777
- Lista é uma cópia de lst

7. A memória do computador funciona, basicamente, armazenando sequências de bytes, e cada sequência tem seu próprio tipo e endereço de memória. A quantidade de bytes de uma sequência é determinada pelo tipo, que pode ser: tipo primitivo, ponteiro, estrutura de dados ou a combinação ilimitada destes. O endereço de memória é representado por um número hexadecimal na forma 0x1234.

Com base nestas informações, avalie as afirmações a seguir.

I. Para obter o endereço de memória de uma variável qualquer, utiliza-se o operador "&" antes do nome dessa variável.

II. O que um ponteiro carrega dentro dele é sempre um número hexadecimal, independentemente do tipo de dado para o qual ele aponta.

III. Para obter o valor que está no endereço de memória que está dentro de uma variável do tipo ponteiro, utiliza-se o operador "*" antes do nome da variável ponteiro.

IV. Para obter o conteúdo de uma parte de uma estrutura de dados, referenciada por uma variável do tipo desta estrutura, utiliza-se o operador "." (ponto) depois do nome da variável, seguido do nome da variável da parte desejada.

É correto apenas o que se afirma em

A) I, II e IV.

B) I, II, III e IV.

C) II e III.

D) I, III e IV.

E) I e IV.

8. Este código foi executado e produziu a saída da console.

```c
#include    <stdio.h>
#include    <stdlib.h>

void funcaoA (int ** pt_ptL, int * ptLista, int L) {

    printf("&pt_ptL = %x\npt_ptL = %x\n*pt_ptL = %x\n**pt_ptL = %d\n",
            &pt_ptL, pt_ptL, *pt_ptL, **pt_ptL);
}

int main ()
{
    int l = 77777;

    int * pl = (int *) malloc (sizeof(int));

    *pl = l;

    funcaoA (&pl, pl, l);
}
```

CONSOLE:

```
&pt_ptL = 62fdf0
pt_ptL = 62fe10
*pt_ptL = 151480
**pt_ptL = 77777

--------------------------------
Process exited after 0.0986 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Preencha a tabela abaixo com os valores correspondentes às variáveis.

	l	pl	pt_ptL	pt_Lista	L
Tipo					
Endereço					
Valor					

9. Modifique o código abaixo para coletar do teclado a quantidade de números a serem digitados e armazenados, faça a alocação dinâmica do array com a dimensão informada.

```c
#include    <stdio.h>
#include    <locale.h>

int main()
{
    setlocale(0, "Portuguese");

    int     nros[5], i;

    // entrada - capturar um número e guardar em um array, repetindo 5 vezes
    for (i = 0; i < 5; i++) {
        printf("Digite um número: ");
        scanf("%d", &nros[i]);
    }

    // saida - escrever número por número, todos lidos na entrada
    // na ordem crescente dos índices -> de 0 até 4
    for (i = 0; i < 5; i++)
        printf("%d\t", nros[i]);

    printf("\n");

    // na ordem descrescente dos índices -> 4 até 0
    for (i = 4; i >= 0; i--)
        printf("%d\t", nros[i]);
}
```

10. Escreva um programa em C, com alocação de memória totalmente dinâmica, para armazenar *strings* em uma matriz. Você deve capturar do teclado o tamanho da maior *string* a ser armazenada e as dimensões da matriz. Por exemplo, a maior *string* terá 124 caracteres e a matriz será de 4x5.

11. Escreva um programa em C, totalmente dinâmico, para realizar as quatro operações (CRUD) de um cadastro de alunos: **C**reate, **R**ead, **U**pdate e **D**elete. Cada operação deverá estar em uma função. Deverá ter uma função para mostra o cadastro inteiro. O cadastro deverá armazenar **nome completo, RGM e curso**. O nome deverá ser coletado do teclado e armazenado em uma área estática temporária de tamanho 1024 caracteres. Após ter o nome, alocar a área exata que irá guardar o nome. Os cursos, listados abaixo, deverão estar em um ENUMERATION, sendo que o valor armazenado será o número inteiro, mas na hora de mostrar, deverá mostrar o nome do curso.

Técnicas de Desenvolvimento de Algoritmos
Prof. Walace Bonfim

1 - Análise e Desenvolvimento de Sistemas

2 - Ciência da Computação

3 - Ciência de Dados

4 - Gestão da Tecnologia da Informação

5 - Redes de Computadores

6 - Sistemas Para Internet