Faculdade de Computação (FACOM) - UFMS
Algoritmos e Programação 2
Prof. Dr. Anderson Bessa da Costa

Lista 4 - Ponteiros

1. Considere o seguinte trecho de código:

```
1  #include <stdio.h>
2
3  int main() {
4      int *p, a;
5
6      a = 10;
7      p = &a;
8      *p = *p * 2;
9      printf("a = %d\n", a);
10
11     return 0;
12 }
```

   Qual o valor da variável `a` ao final da execução? (Tente primeiro responder apenas analisando o código. Confirme a sua resposta compilando e executando o código.).

2. Considere o seguinte trecho de código:

```
1  #include <stdio.h>
2
3  int main() {
4      float *q, b;
5
6      b = 3.14;
7      *q = *q + 1;
8      printf("*q = %f\n", *q);
9
10     return 0;
11 }
```

   Onde está o erro neste código? O que acontecerá se você executá-lo desta maneira?

3. Declare duas variáveis `x` e `y` do tipo `int` e duas variáveis `p` e `q` do tipo `int*`. Atribua o valor 2 para `x` e 8 para `y`, o endereço de `x` para `p` e o endereço de `y` para `q`. Então, imprima (usando `printf`) as seguintes informações:

   (a) O endereço de `x` e o valor de `x`;

   (b) O valor de `p` e o valor de `*p`;

   (c) O endereço de `y` e o valor de `y`;

   (d) O endereço de `p` (não o seu conteúdo!).

   Obs.: Um endereço deve ser impresso utilizando o especificador de formato `%p` no `printf`.

4. Discuta, passo a passo, o efeito do seguinte fragmento de código:

```
1  int *v;
2  v = (int *) malloc (10 * sizeof(int));
```

5. Escreva um programa que solicite ao usuário o número de notas a serem fornecidas e aloque dinamicamente um vetor com a dimensão especificada para armazenar as entradas. Solicite as notas e chame uma função que retorne a média aritmética das notas. Após imprimir a média o programa deve liberar a memória dinâmica que fora alocada.

   Se você já viu o modificador `const`, o utilize no parâmetro que recebe as notas, a fim de garantir que o seu conteúdo não será alterado.

6. Implemente a função `troque()` que faz uso da passagem por valor. Essa função deve realizar a troca dos argumentos `a` e `b`. Teste na `main` e veja que a função implementada não funciona. Explique.

7. Implemente a função `troque2()` que faz uso da passagem por referência com argumentos de referência (específico de C++, utilizando o operador de referência & a frente dos parâmetros). Essa função deve realizar a troca dos argumentos `a` e `b` passados por referência. Teste na `main` se a sua função está funcionando corretamente.

8. Implemente a função `troque3()` que faz uso da passagem por referência com argumentos de ponteiro. Esta função deve realizar a troca dos argumentos `a` e `b` passados por referência (passa-se o endereço neste caso específico). Teste na `main` se a sua função está funcionando corretamente.

9. Escreva um programa que recebe três variáveis do tipo int `(a, b, c)` e rotacione os valores de forma que o valor de a vai para b, o valor de b para c e o valor de c para a. Utilize chamadas a função `troque` desenvolvida anteriormente (escolha uma das duas implementações da função que funciona: `troque1` ou `troque2`).

10. Crie uma função chamada `print_addr(int x)` cujo propósito seja apenas imprimir o conteúdo da variável x e seu endereço. Declare uma variável x do tipo `int` na main e inicialize-a com o valor 3. Imprima seu conteúdo e endereço dentro da main. Então, chame a função `print_addr` passando a variável x como argumento. Compare os resultados. Este comportamento é o esperado?

11. O seguinte trecho de código calcula a soma dos n primeiros números naturais.

```
1  #include <stdio.h>
2
3  int main () {
4    int n, soma;
5
6    soma = 0;
7
8    scanf("%d", &n);
9
10   while (n > 0) {
11     soma = soma + n;
12     n -= 1;
13   }
14
15   printf("A soma dos %d primeiros numeros e: %d", n, soma);
16
17   return 0;
18 }
```

Reescreva o código utilizando duas funções. Uma função deve calcular a soma utilizando passagem por referência e a segunda função exibe o resultado utilizando passagem por valor.

12. Crie um programa que use ponteiro para função a fim de executar diferentes operações aritméticas baseado na entrada do usuário.

    Crie quatro funções simples: `adicione`, `subtraia`, `multiplique` e `divida`. Cada função deve receber dois argumentos `int` e retornar um `int` como resultado. A função `divida` deve tratar divisão por zero a fim de prevenir erros em tempo de execução.

    Declare um ponteiro para função que possa apontar para qualquer uma dessas quatro funções. A assinatura do ponteiro deve casar com as assinaturas das funções (ou seja, `int (*ptr) (int, int)`).

    Entrada:

    - Solicite ao usuário entrar com dois inteiros;
    - Solicite ao usuário escolher uma operação (por exemplo, `'a'` para adicionar, `'s'` para subtrair, `'m'` para multiplicar e `'d'` para dividir).

    Atribuir e chamar:

    - Baseado na escolha do usuario, atribua o endereço da função correspondente para o ponteiro para função;
    - Chame a função escolhida usando ponteiro para função, passando os dois inteiros de entrada como argumentos;
    - Imprima o resultado da operação.