Estrutura de Dados
Exercícios: *ponteiros*

1 Ambiente de desenvolvimento

1. Baixe, instale e configure um compilador C, juntamente com o editor ou IDE de sua preferência, de forma a já ficar desde o início com um ambiente de desenvolvimento devidamente pronto para o restante do curso.
2. Veja na sala do Moodle os *links* dos programas para Windows e Linux.

2 Regras

1. Desenvolva os exercícios abaixo em C, utilizando como referência as informações apresentadas na aula e a apostila de C que se encontra no Moodle.
2. Para cada exercício, crie um arquivo com o nome `exN.c`, onde `N` é o número do exercício.
3. Sempre que o exercício pedir para se criar uma função, crie um programa cliente correspondente para testá-la.

3 Exercícios

1. Implemente a função com o cabeçalho

   ```
   void calcula(float l, float *area, float *perimetro);
   ```

   que recebe o lado `l` de um quadrado e retorna a sua área e perímetro.

2. Crie uma função que imprime um vetor de inteiros de tamanho arbitrário, com o seguinte cabeçalho:

   ```
   void imprime(int *v, int tam);
   ```

3. Implemente a função com o cabeçalho

   ```
   void swap(int *a, int *b);
   ```

   inverte os valores inteiros apontados por `a` e `b`.

4. Crie um programa que percorre uma matriz de inteiros 5×4 usando apenas um *loop* com ponteiros. Preencha cada posição com o seu valor ordinal, começando do zero. *Exemplo*: `m[0][0] == 0`, `m[0][1] == 1`, `m[1][0] == 4`, etc. *Obs.:* Não é necessário alocar a matriz dinamicamente, faça tudo na função principal.

5. Considere uma versão simplificada da função `memcpy` como abaixo:

   ```
   void memcpy(char* dest, const char* src, int count) {
     while (count--) *dest++ = *src++;
   }
   ```

   Esta função copia `count` *bytes* (`char`) do endereço de fonte `src` para o endereço de destino `dest`, sem considerar sobreposição de áreas de memória. Reescreva a essa função usando a notação de vetor, mas mantendo o mesmo comportamento da função original.