# INF 112 – Prática 1, revisão de ponteiros – 2018/2

**Exercício 1** (salve-o com o nome `exercicio1.cpp`): Implemente duas funções com nome `leiaInteiros`. Quando chamada, cada função deverá pedir para o usuário digitar dois números inteiros e, então, ler tais números a partir do teclado. Faça com que cada função receba 2 parâmetros e grave os dois inteiros lidos nesses parâmetros. Na primeira função, os parâmetros deverão ser passados por referência, enquanto na segunda função eles deverão ser passados utilizando ponteiros.

Implemente também um método `main` e faça com que ele leia dois números utilizando a primeira versão da função `leiaInteiros` e, então, imprima esses números na tela. Após isso, seu método `main` deverá fazer a mesma coisa utilizando a segunda versão da função.

**Exercício 2** (salve-o com o nome exercicio2.cpp): Implemente três funções com nome `calculaTamanhoStringX` (para diferenciá-las, troque *X* por 1, 2 e 3). Suas funções deverão receber como parâmetro um apontador para `char` e, então, contar quantos caracteres a `string` apontada possui. Lembre-se que as `strings` "de C" (representadas por arranjos de caracteres) possuem um caractere de terminação (o caractere `'\0'`).

a) A primeira função deverá utilizar uma estrutura `for` para varrer o arranjo de caracteres de forma "tradicional" (utilizando o `[]`).

b) A segunda função deverá utilizar uma estrutura do tipo `for` que deverá, então, utilizar aritmética de ponteiros (para acessar a posição `i` do arranjo, basta somar `i` ao ponteiro) para acessar os caracteres apontados pelo ponteiro.

c) A terceira função deverá utilizar uma estrutura do tipo `for` que deverá incrementar o ponteiro de modo a se deslocar pelo arranjo de caracteres. Nessa última função, você poderá declarar APENAS variáveis do tipo apontador para char! Além disso, não utilize a notação de arranjo (`[]`) para acessar os dados. Dica: utilize o operador `sizeof()` para saber o número de bytes ocupados por cada caractere.

Utilize a função `main` abaixo para testar suas funções:

```
int main() {
  char str[51];
  cout << "Digite alguma string... (com até 50 caracteres):";
  cin.getline(str, 50);
  cout << "Tamanhos calculados:" << endl;
  cout << calculaTamanhoString1(str) << " " << calculaTamanhoString2(str)
       << " " << calculaTamanhoString3(str);
  cout << endl;
  return 0;
}
```

**Exercício 3** (salve-o com o nome exercicio3.cpp): Considerando o registro abaixo, faça um programa com uma função `main` e uma função `leDadosJogador`. Na função `main`, declare um *array* com 5 jogadores e, em um laço do tipo `for`, chame a função `leDadosJogador` para cada jogador no arranjo e, após isso, varrer o arranjo novamente imprimindo os dados dos jogadores. A função `leDadosJogador` deverá receber como parâmetro um ponteiro para `jogador`, ler a partir do usuário os dados de um `jogador` (pontos, coordenadas `x` e `y`) e, finalmente, gravar tais dados no jogador recebido como parâmetro.

```
struct Jogador {
    int pontos;
    int x,y;
};
```

**Exercício 4** (salve-o com o nome exercicio4.cpp): Complete o trecho de código abaixo de modo que ele imprima a `string` em ordem reversa. ATENÇÃO: Não utilize a notação `[]` no trecho que você completar. Utilize apenas operações sobre apontadores!.

```
int main() {
    char str[] = "abc teste";

    //Insira o codigo aqui...

    //

    return 0;
}
```

**Exercício 5** : Uma importante ferramenta presente no Linux (e em outros Sistemas Operacionais) é o Make, que facilita a tarefa de compilação dos programas. Estude o uso do Make e desenvolva um Makefile capaz de compilar todos os programas desenvolvidos nesta aula.