



Pontifícia Universidade Católica de Minas Gerais

ICEI – Instituto de Ciências Exatas e Informática
DCC – Departamento de Ciência da Computação
Campus Belo Horizonte – Coração Eucarístico
Bacharelado em Ciência da Computação
Algoritmos e Estruturas de Dados I

MAIOR UNIVERSIDADE CATÓLICA DO MUNDO - Fonte: Vaticano
MELHOR UNIVERSIDADE PRIVADA DO BRASIL – 6x pelo Guia do Estudante
COMPUTAÇÃO PUC MINAS: 2º LUGAR DO BRASIL (Pref. Mercado) – Folha de São Paulo, 2018
CIÊNCIA DA COMPUTAÇÃO PUC MINAS: 5 ESTRELAS - Guia do Estudante, 2018
CIÊNCIA DA COMPUTAÇÃO PUC MINAS: NOTA MÁXIMA NO MEC - Conceito 5 no último ENADE

Professor: Lúcio Mauro Pereira
13 de maio de 2019
Lista de Exercícios nº 16

Arranjos unidimensionais

Estudar:

Obra: Fundamentos da Programação de Computadores. Autora: Ana Ascêncio

Disponível na biblioteca da PUC Minas de forma física e *e-book*.

Capítulo 6: Vetor

Obra: C: como programar. 8ed. Autor: Deitel.

Disponível na biblioteca da PUC Minas de forma física e *e-book*.

Capítulo 6: Arrays

1. Construa uma função que troque o primeiro elemento do arranjo com o último.
Argumento: um arranjo de inteiros.
Teste o método criado a partir de `main()`.
2. Construa uma função que troque dois elementos quaisquer de um arranjo de reais.
Argumentos: Um arranjo de reais e dois valores inteiros relativos às duas posições a serem trocadas.
Teste o método criado a partir de `main()`.
3. Considere a função `main()` abaixo:

```
int main() {  
    int A[] = {6, 5, 4, 3, 2, 1};  
    int B[] = {3, 1, 6, 4, 2, 5};  
    int C[6];  
    leArranjo(C); // Preenche todo o arranjo com valores lidos  
    deslocaMaiorParaFinal(A); // Desloca o maior valor do arranjo para a última posição  
    deslocaMaiorParaFinal(B);  
    deslocaMaiorParaFinal(C);  
    escreveArranjo(A); // Escreve na tela todo o arranjo  
    escreveArranjo(B);  
    escreveArranjo(C);  
    return 0;  
}
```

Implemente os métodos evocados na função principal acima.

Obs: Busque alguma estratégia que não permita o maior valor sobrepor aquele inicialmente posicionado na última posição do arranjo.

4. Construa uma nova versão do programa acima, substituindo o deslocamento do maior valor pela ordenação completa do arranjo em forma crescente.

```
int main() {  
    int A[] = {6, 5, 4, 3, 2, 1};  
    int B[] = {3, 1, 6, 4, 2, 5};  
    int C[6];  
    leArranjo(C); // Preenche todo o arranjo com valores lidos  
    ordena(A); // Desloca o maior valor do arranjo para a última posição, n vezes, até ordenar  
    ordena (B);  
    ordena (C);  
    escreveArranjo(A); // Escreve na tela todo o arranjo  
    escreveArranjo(B);  
    escreveArranjo(C);  
    return 0;  
}
```

5. Construa um programa que calcule e escreva o número de alunos em uma turma com idade superior à idade média da turma.
A leitura das idades deverá se repetir até que uma idade igual a zero seja lida (*flag*).
Rejeitar a leitura de idades inválidas. Considere o domínio: 0..150
Um método deverá ficar encarregado de ler uma idade.
Um método deverá ficar encarregado de calcular a idade média da turma (este deve chamar a soma).
Um método deverá ficar encarregado de calcular o número de alunos com idade acima da média das idades da turma (este deve chamar o método que calcula a média).
Fique à vontade para projetar outros métodos auxiliares.
6. Construa um método que receba um arranjo de inteiros e preencha-o com a série de Fibonacci.
Observe que o arranjo lhe permite criar uma interessante versão iterativa (não use recursividade).
Teste sua solução a partir do método principal.
7. Construa um método que conte o número de elementos negativos em um arranjo de reais.
Teste o método criado a partir do método principal.
8. Desafio: Planeje e implemente uma versão recursiva para o problema acima.