Programação 2

Aula 1

Thiago Cavalcante – thiago.kun@gmail.com

23 de outubro de 2019

Universidade Federal de Alagoas – UFAL
Campus Arapiraca
Unidade de Ensino de Penedo

Sumário

1. Introdução
    - Informações Gerais da Disciplina
    - Pesquisa
    - Recapitulando: Programação 1
    - Ementa
    - Bibliografia
    - Cronograma

2. Estruturas de Dados

Introdução

Informações Gerais da Disciplina

Nome	Programação 2
Código	SISB087
Semestre	3º
Carga Horária	72h
PPC	02/2019
Turma	2019.2
Horário	Quarta, 19:00 – 22:30
Sala	5
Lista de discussão	Google Groups: sisb087_20192
Repositório	GitHub: theagoliveira/sisb087_20192

Pesquisa

1. Pode trazer um computador para aulas práticas?
2. Já usou **Git**?
3. Já usou **GitHub**?
4. Consegue compreender textos em **inglês**?

```
<NOME COMPLETO> <E-MAIL> <RESPOSTAS>
```

Recapitulando: Programação 1

- Linguagem: **C**
- Conceitos e técnicas de programação básica, procedimentos, algoritmos e programas
- Identificadores, constantes, variáveis e atribuição
- Tipos primitivos de dados
- Comandos de entrada e saída
- Operadores, funções e expressões
- Instruções condicionais e de repetição
- Tipos definidos pelo programador e tipos abstratos de dados
- Noções de ponteiros
- Estruturas compostas de dados: vetores, matrizes e registros
- Manipulação de uma cadeia de caracteres
- Noções de arquivos
- Programação de algoritmos usando uma LP estruturada
- Boas práticas de programação

Ementa

- Linguagem: **C**
- Importância das estruturas de dados na solução de problemas
- Vetores e matrizes
- Estruturas de dados lineares e não lineares
- Pilhas, filas, listas, árvores, florestas, introdução à grafos
- Implementação de estruturas de dados com alocação estática e dinâmica de memória
- Implementação de estruturas de dados com e sem ponteiros
- Algoritmos de ordenação
- Algoritmos de busca
- Programação avançada e resolução de problemas complexos
- Introdução à análise de algoritmos

Ementa

- Linguagem: **C**
- Importância das estruturas de dados na solução de problemas
- ~~Vetores e matrizes~~
- Estruturas de dados lineares e não lineares
- Pilhas, filas, listas, árvores, florestas, introdução à grafos
- Implementação de estruturas de dados com alocação estática e dinâmica de memória
- Implementação de estruturas de dados com e sem ponteiros
- Algoritmos de ordenação
- Algoritmos de busca
- Programação avançada e resolução de problemas complexos
- Introdução à análise de algoritmos

Livros sugeridos no repositório

- Programação em C
- Estruturas de dados com exemplos em C
- Estruturas de dados com exemplos em outras linguagens

Livros sugeridos no repositório

- Programação em C
- Estruturas de dados com exemplos em C
- Estruturas de dados com exemplos em outras linguagens

Cronograma

Datas importantes	
13/11/2019	*Possível* dia da AB1
20/11/2019	Feriado
23/11/2019	Prazo final para digitação da AB1
12/02/2020	*Possível* dia da AB2
17/02/2020	Prazo final para digitação da AB2
17–22/02/2020	Período de reavaliação
27–29/02/2020	Período de provas finais

Estruturas de Dados

Introdução a Estruturas de Dados

$$\textbf{Programa} = \textbf{Algoritmos} + \textbf{Estruturas de Dados}$$

Programa $=$ **Algoritmos** $+$ **Estruturas de Dados**

Escolha e implementação da estrutura de dados **são tão importantes quanto** as rotinas

$$\textbf{Programa} = \textbf{Algoritmos} + \textbf{Estruturas de Dados}$$

Escolha e implementação da estrutura de dados **são tão importantes quanto** as rotinas

O tipo de estrutura é **determinado pela natureza** do problema

Tipos abstratos de dados podem ser implementados **corretamente** com diferentes estruturas de dados

Introdução a Estruturas de Dados

Tipos abstratos de dados podem ser implementados **corretamente** com diferentes estruturas de dados

A forma como as operações são realizadas por uma estrutura podem **melhorar** (ou **piorar**) drasticamente a performance de um programa

Introdução a Estruturas de Dados

Tipos abstratos de dados podem ser implementados **corretamente** com diferentes estruturas de dados

A forma como as operações são realizadas por uma estrutura podem **melhorar** (ou **piorar**) drasticamente a performance de um programa

Projeto: melhor prevenir do que remediar

Três motivos para usar estruturas de dados:

- **Eficiência**: a forma de organização dos dados pode aumentar a velocidade de execução de algoritmos (de busca e/ou ordenação, por exemplo)

Três motivos para usar estruturas de dados:

- **Eficiência**: a forma de organização dos dados pode aumentar a velocidade de execução de algoritmos (de busca e/ou ordenação, por exemplo)
- **Abstração**: melhor entendimento dos dados na solução de problemas

Três motivos para usar estruturas de dados:

- **Eficiência**: a forma de organização dos dados pode aumentar a velocidade de execução de algoritmos (de busca e/ou ordenação, por exemplo)
- **Abstração**: melhor entendimento dos dados na solução de problemas
- **Reusabilidade**: estruturas de dados são modulares e independentes de contexto

Introdução a Estruturas de Dados

Características:

- Estrutura Linear x Não-linear

Introdução a Estruturas de Dados

Características:

- Estrutura Linear x Não-linear
- Estrutura Contígua x Ligada

Introdução a Estruturas de Dados

Características:

- Estrutura Linear x Não-linear
- Estrutura Contígua x Ligada
- Dados Homogêneos x Heterogêneos

Introdução a Estruturas de Dados

Características:

- Estrutura Linear x Não-linear
- Estrutura Contígua x Ligada
- Dados Homogêneos x Heterogêneos
- Alocação de Memória Estática x Dinâmica

Tipos Abstratos de Dados – TAD

”Um TAD pode ser visto como um modelo matemático, acompanhado das operações definidas sobre o modelo.”

Tipos Abstratos de Dados – TAD

”Um TAD é um tipo de dados (conjunto de valores e operações sobre esses valores) que é acessado apenas através de uma interface.”

Tipos Abstratos de Dados – TAD

”Nos casos em que um módulo define um novo tipo de dado e o conjunto de operações para manipular dados desse tipo, dizemos que o módulo representa um TAD. Nesse contexto, abstrato significa ’esquecida a forma de implementação’, ou seja, um TAD é descrito pela finalidade do tipo e de suas operações, e não pela forma como está implementado.”

Tipos Abstratos de Dados – TAD

Exemplos:

- TAD Ponto
- TAD Círculo

Exercício: TAD Matriz

```
typedef struct matriz Matriz;

Matriz* mat_cria(int m, int n);
void mat_libera(Matriz* mat);
float mat_acessa(Matriz* mat, int i, int j);
void mat_atribui(Matriz* mat, int i, int j, float v);
int mat_linhas(Matriz* mat);
int mat_colunas(Matriz* mat);

struct matriz {
    int lin;
    int col;
    float* v;
    /* outra opção: matriz como vetor de ponteiros */
    /* float** v; */
}
```

Dúvidas?