Estrutura de Dados

Estruturas e Ponteiros

Prof. Dr. Daniel Caetano

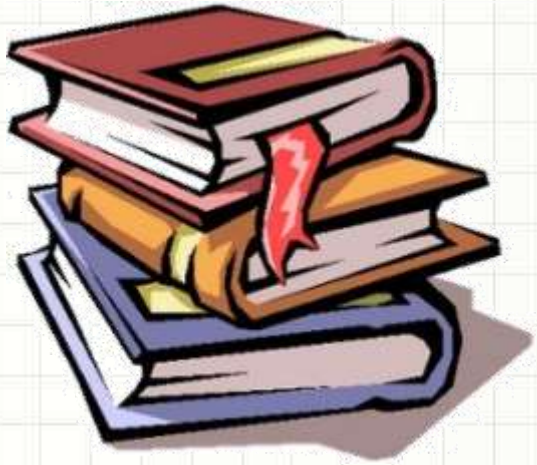
2012 - 2

Objetivos

- Compreender o que são estruturas
- Compreender sua aplicação
- Compreender o que são ponteiros
- Capacitar para implementar programas com estruturas

- **Atividade Estruturada!**

Material de Estudo



Material	Acesso ao Material
Apresentação	http://www.caetano.eng.br/ (Aula 9)
Material Didático	Estruturas de Dados (Parte 1) – Páginas 36 a 41

RECORDANDO...

Recordando...

- Vimos...
  - Listas Ordenadas / Não Ordenadas
  - Pilhas
  - Filas
- Só armazenar dados desagregados?
- E se quiser armazenar um **aluno**?
  - Matricula - int
  - Nota - float
- Usar dois vetores?

Tipos Abstratos de Dados

Tipos Abstratos de Dados

- Linguagem traz alguns tipos de dados
  - int, float, char etc.
- E se precisarmos de um diferente?
  - Data: dia, mês, ano
  - Cliente: cpf, nome, cartão de crédito
  - Aluno: matrícula e nota
- Como fazer?
  - Criar um tipo abstrato de dado

Tipos Abstratos de Dados

- Struct

```
struct aluno {
        int matricula;
        float nota;
        };
```

- Como usar isso?

```
struct aluno umAluno;
umAluno.matricula = 103567;
umAluno.nota = 7.5;
```

Tipos Abstratos de Dados

- Acompanhe o exemplo:
  - Implementanto um tipo de dado “aluno”.

Tipos Abstratos de Dados

- Acompanhe outro exemplo:
  - Passando um “aluno” como parâmetro de função.

Tipos Abstratos de Dados

- Acompanhe outro exemplo:
  - Vetores de “alunos”.

Tipos Abstratos de Dados

- Acompanhe outro exemplo:
  - Passando vetores de “alunos” como parâmetro.

Tipos Abstratos de Dados

- Acompanhe outro exemplo:
  - Usando “alunos” em estruturas de dados...
  - Observe que podemos “copiar” o conteúdo de um aluno para outro simplesmente assim:

```
struct aluno umAluno, outroAluno;
umAluno.matricula = 103567;
umAluno.nota = 7.5;
outroAluno = umAluno;
```

Endereços e Ponteiros

Endereços e Ponteiros

- Onde fica armazenado o valor da variável?
  - Memória!
- Como é a memória do computador?
  - Um monte de “gavetas”: **posição de memória**
  - Cada gaveta tem um número único: **endereço**
- Cada nome de variável → endereço
  - Declarar variável =?
  - Armazenar valor em variável = ?

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome								
Valor	??	??	??	??	??	??	??	??

- char letra;

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra							
Valor	??	??	??	??	??	??	??	??

- char letra;
- int idade;

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra	idade						
Valor	??	??	??	??	??	??	??	??

- char letra;
- int idade;
- char a[3];

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra	idade				a		
Valor	??	??	??	??	??	??	??	??

- char letra;
- int idade;
- char a[3];
- letra = ‘c’;

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra	idade				a		
Valor	99	??	??	??	??	??	??	??

- char letra;
- int idade;
- char a[3];
- letra = ‘c’;
- A[1] = ‘d’;

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra	idade				a		
Valor	99	??	??	??	??	??	100	??

- char letra;
- int idade;
- char a[3];
- letra = ‘c’;
- A[1] = ‘d’;
- idade = 256;

Endereços e Ponteiros

- Cada nome de variável → endereço

Endereço	0	1	2	3	4	5	6	7
Nome	letra	idade				a		
Valor	99	0	1	0	0	??	100	??

- char letra;
- int idade;
- char a[3];
- letra = ‘c’;
- A[1] = ‘d’;
- idade = 256;

Como saber o endereço de uma variável?

Prefixo &

Endereços e Ponteiros

- Exemplo

```
int a=1, b=2;
cout << "Val.A: " << a <<endl;
cout << "End.A: " << &a <<endl;
cout << "Val.B: " << b <<endl;
cout << "End.B: " << &b <<endl;
```

Endereços e Ponteiros

- Podemos guardar endereço em uma variável?

```
int a=1, b;
b = &a;
cout << "End.A: " << &a <<endl;
cout << "Val.B: " << b <<endl;
```

- Funciona?
- Precisamos de um tipo novo de variável...

Ponteiros

Ponteiros

- Ponteiro: serve para armazenar um endereço
  - Indicado por um * na frente do nome da variável

```
int a=1, *b;
b = &a;
cout << "End.A: " << &a <<endl;
cout << "Val.B: " << b <<endl;
```

- E se quisermos saber o valor da variável "**apontada**"?
  - Usamos o * na frente do nome do ponteiro

Ponteiros

- Exemplo: Lendo valor apontado

```
int a=1, *b;
b = &a;
cout << "Val.A: " << a <<endl;
cout << "End.A: " << &a <<endl;
cout << "Val.B: " << b <<endl;
cout << "Val.*B: " << *b <<endl;
```

Ponteiros

- Acompanhe o exemplo:
  - Conhecendo o básico sobre ponteiros.

Ponteiros

- Acompanhe outro exemplo:
  - Mudando valores usando ponteiros.

Ponteiros

- Acompanhe outro exemplo:
  - Ponteiros nulos: NULL.

Ponteiros

- Acompanhe outro exemplo:
  - Ponteiros para estruturas.

Ponteiros

- Acompanhe outro exemplo:
  - Ponteiros para estruturas e o referenciador **->** .

EXERCÍCIOS DE FIXAÇÃO

Exercício 1

- Baixe a “base” da aula de hoje e, com base no arquivo exercicio1, modifique o programa para que apenas alunos com nota igual ou superior a 6.0 tenham sua matrícula impressa.

Exercício 2

- Baixe a “base” da aula de hoje e, com base no arquivo exercicio2, modifique o programa para que ele empilhe as cartas do baralho e depois desempilhe-as.

Conclusões

Resumo

- Tipos Abstratos de Dados (structs) tornam as estruturas de dados muito mais poderosas
- Além do uso normal das variáveis, podemos verificar e usar seus endereços...
- Que serão importantes para criar estruturas de dados de tamanho indeterminado!

- **TAREFA**
  - **Atividade Estruturada!**

Próxima Aula

- Como usar esse conceitos para aplicativos mais úteis?

PERGUNTAS?

Bom Descanso a Todos!