# Classificação dos tipos de dados

- Tipos de Dados Simples
  - Tipos de Dados Inteiros: *int*, *long* e *unsigned int*
  - Tipos de Dados Reais: *float* e *double*
  - Tipos de Dados Caracteres: *char*
- Tipos de Dados Estruturados
  - cadeia de caracteres *(string)*, vetores *(array)*, registos *(struct)* e arquivos em disco.
- ponteiros (alocação dinâmica de memória)
- Classes e Objetos (Orientação a Objetos)

# Tipos de Dados Estruturados

- armazenam **diversos** itens de uma só vez

- isto significa:
  - em uma mesma estrutura de dados, é possível ter diversas variáveis de tipos de dados simples agrupadas

# Lembrar Vetores

- representa um conjunto de valores do mesmo tipo (estrutura homogênea), referenciáveis pelo mesmo nome e individualizados entre si através de sua posição dentro desse conjunto (variáveis indexadas)
- portanto, os vetores, armazenam "múltiplos" valores do mesmo tipo

---

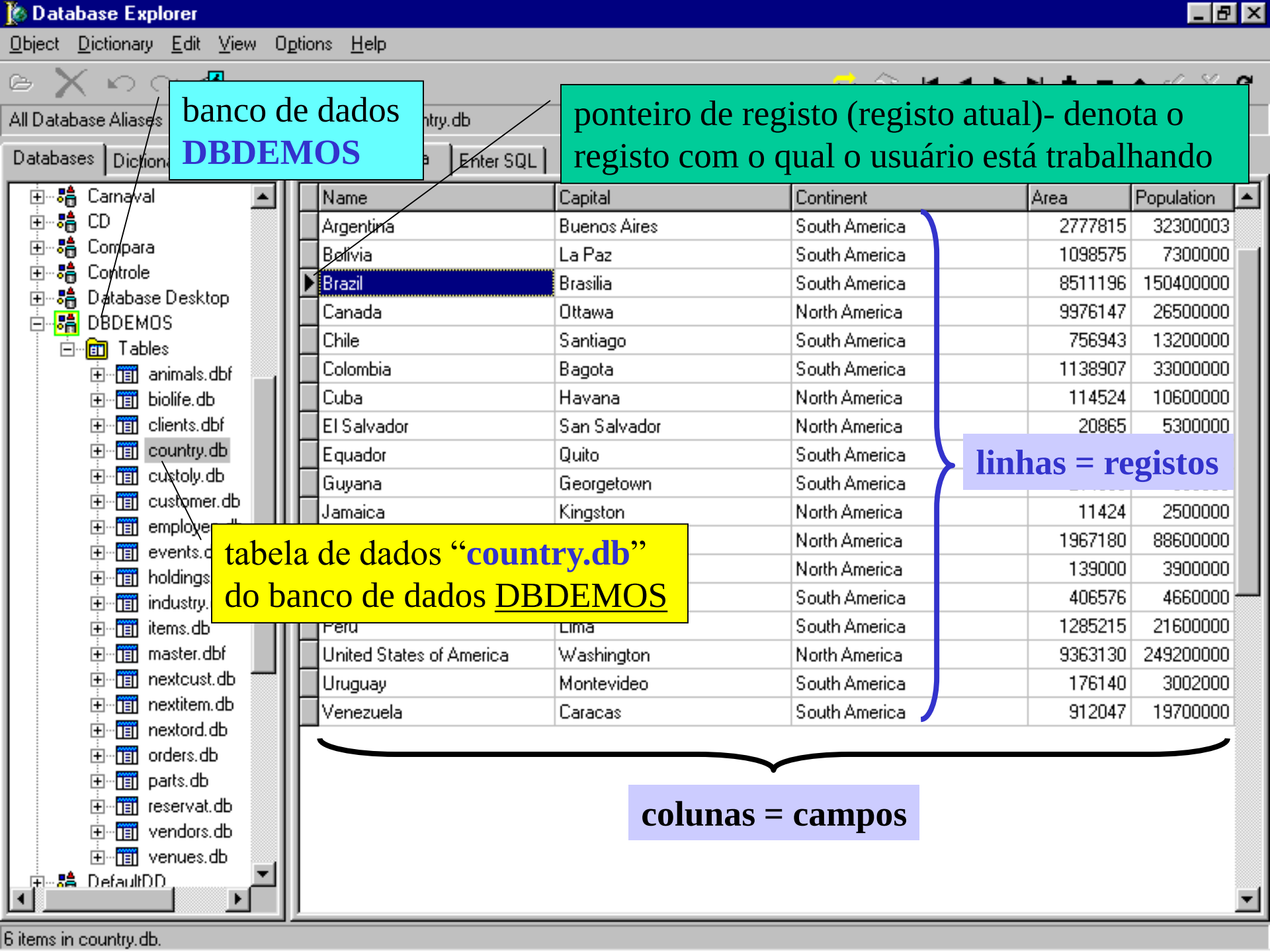
- para armazenar, ou agrupar, informações "relacionadas" que têm tipos diferentes deve-se utilizar variáveis do tipo registo, ou estrutura (**struct**)

# Definições de registo (**struct**)

- representa um conjunto de valores logicamente relacionados, que podem ser de tipos diferentes (**estrutura heterogénea**)
- junção ou composição de tipos em um tipo composto
- conjunto de informações agrupadas e relacionadas entre si
- agrupamento de variáveis, não necessariamente do mesmo tipo, que guardam estreita relação lógica

# ***struct***, características básicas

- contém um número fixo de elementos chamados de campos, ou “membros”
- os campos podem ser de tipos diferentes (estrutura heterogênea)
- cada campo tem um nome próprio chamado de “*identificador de campo*”
- campo = unidade de registo


Database Explorer
Object Dictionary Edit View Options Help
All Database Aliases
Databases
Enter SQL
Carnaval
CD
Compara
Controle
Database Desktop
DBDEMOS
Tables
animals.dbf
biolife.db
clients.dbf
country.db
custoly.db
customer.db
employee.db
events.db
holdings.db
industry.db
items.db
master.dbf
nextcust.db
nextitem.db
nextord.db
orders.db
parts.db
reservat.db
vendors.db
venues.db
DefaultDD
Name
Capital
Continent
Area
Population
Argentina Buenos Aires South America 2777815 32300003
Bolivia La Paz South America 1098575 7300000
Brazil Brasilia South America 8511196 150400000
Canada Ottawa North America 9976147 26500000
Chile Santiago South America 756943 13200000
Colombia Bagota South America 1138907 33000000
Cuba Havana North America 114524 10600000
El Salvador San Salvador North America 20865 5300000
Equador Quito South America
Guyana Georgetown South America
Jamaica Kingston North America 11424 2500000
North America 1967180 88600000
North America 139000 3900000
South America 406576 4660000
Peru Lima South America 1285215 21600000
United States of America Washington North America 9363130 249200000
Uruguay Montevideo South America 176140 3002000
Venezuela Caracas South America 912047 19700000
6 items in country.db.
banco de dados DBDEMOS
ponteiro de registo (registo atual)- denota o registo com o qual o usuário está trabalhando
tabela de dados "country.db" do banco de dados DBDEMOS
linhas = registos
colunas = campos

Lista dos tipos e seus **campos** (ou membros):

```
// definição do tipo registo
struct <IdentificadorDoregisto>
{
  <tipo1> <campo1>;
  <tipo2> <campo2>;
  ...
  <tipoN> <campoN>;
};
```

```
// declaração da variável do tipo registo definido
struct <IdentificadorDoregisto> <NomeDaVariável>;
```

**onde**:

**campo1**, **campo2**, ..., **campoN**: representam os nomes associados a cada campo do registo;

**tipo1**, **tipo2**, ..., **tipoN**: representam qualquer um dos tipos básicos ou 'tipo anteriormente definido'.

Tomando como exemplo a proposta de se criar um registo denominado **rgAluno**, cujos campos são **nome**, **nota1**, **nota2**, **nota3** e **nota4**, este seria assim declarado em C:

```
// definição do tipo registo rgAluno
struct rgAluno {
   char nome[35];
   float nota1;
   float nota2;
   float nota3;
   float nota4;
};
```

```
// declaração da variável Aluno declarada a partir
// do tipo registo rgAluno
struct rgAluno Aluno;
```

Este exemplo corresponde à definição de um modelo de um registo (**rgAluno**) e à criação de uma área de memória chamada **Aluno**, capaz de conter cinco subdivisões, ou campos: **nome**, **nota1**, **nota2**, **nota3** e **nota4**.

Para utilizar um campo específico do registo, deve-se diferenciar esse campo. Para tal utiliza-se o caractere "." (ponto) para estabelecer a separação do nome da variável registo do nome do campo.

```
struct rgAluno {
   char nome[35];
   float nota1;
   float nota2;
   float nota3;
   float nota4;
};

struct rgAluno Aluno;
// referência aos campos da variável registo
   Aluno.nome = "carla";
   Aluno.nota1= 11;
```

**para atribuir tipos estruturados registo (struct)**

Ler valores para os campos do registo:

```
struct rgAluno {
   char nome[35];
   float nota1;
   float nota2;
};
struct rgAluno Aluno;
```

**para atribuir tipos estruturados registo (struct)**

```
void main() {
   printf("Nome do Aluno: ");
   gets(Aluno.nome);

   printf("Nota 1o. Bimestre: ");
   scanf("%f", &Aluno.nota1);

   printf("Nota 2o. Bimestre: ");
   scanf("%f", &Aluno.nota2);
}
```

Escrever os valores dos campos do registo:

```
struct rgAluno {
   char nome[35];
   float nota1;
   float nota2;
};

struct rgAluno Aluno;
```

---

```
void main() {

   ...

   printf("Nome do Aluno: %s\n", Aluno.nome);

   printf("Nota 1o. Bimestre: %5.2f\n", Aluno.nota1);

   printf("Nota 2o. Bimestre: %5.2f\n", Aluno.nota2);

}
```

Escrevendo os valores dos campos do registo:

```
struct rgAluno {
   char nome[35];
   float nota1;
   float nota2;
};

struct rgAluno Aluno;
```

.....

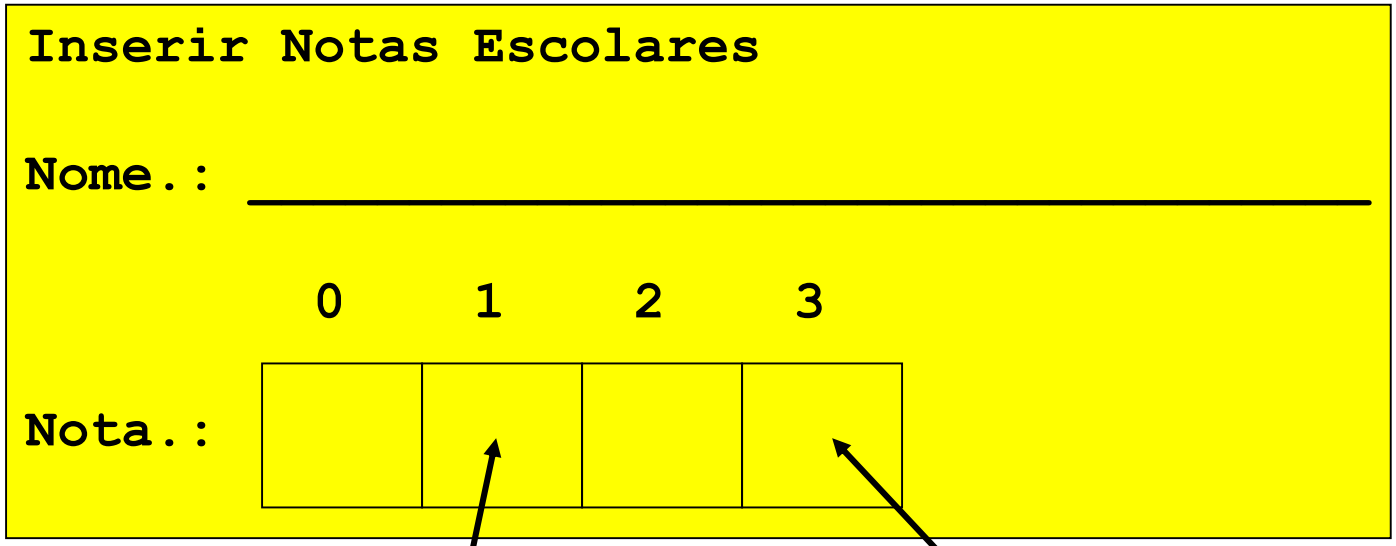
```
void main() {

   ...

   printf("Nome do Aluno: %s\n", Aluno.nome);


}
```

Campos do registo declarados como vetor:



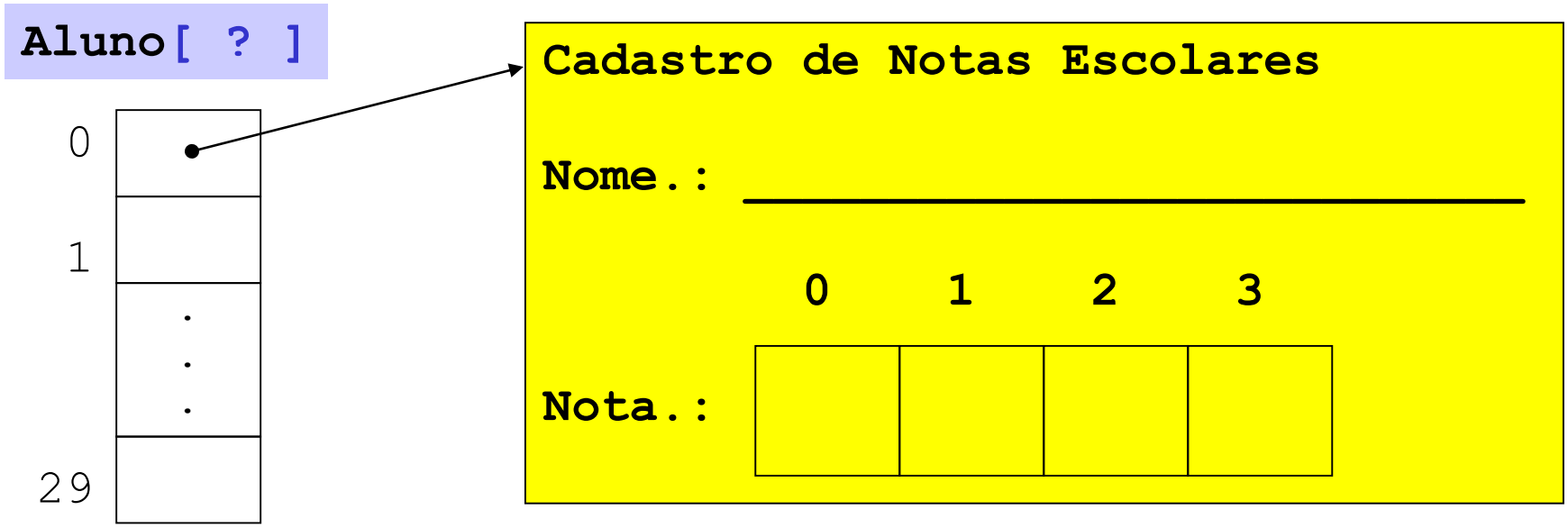

```
struct rgAluno {
   char nome[35];
   float nota[4];
};
struct rgAluno Aluno;
```

Para manipular um campo do registo do tipo vetor deve-se obedecer às manipulações próprias de cada estrutura de dados, ou seja:

```
Aluno.nota[1] = 7.5;     scanf("%f", &Aluno.nota[3]);
```

## Vetores de registos:

Para controlar as notas de 30 alunos, numerados de 0 até 29 sequencialmente, basta criar um vetor no qual cada posição é um elemento do tipo registo **rgAluno**.




```
struct rgAluno {
   char nome[35];
   float nota[4];
};
struct rgAluno Aluno[30];

gets(Aluno[5].nome);      scanf("%f", Aluno[i].nota[1]);
```