ESTRUTURA DE DADOS E ALGORITMOS I

Estrutura de Registros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período

INTRODUÇÃO

- Um vetor é capaz de armazenar diversos valores, com a restrição de que todos sejam de um mesmo tipo de dados.

- Um programa, por exemplo, que gerencie os recursos humanos de uma empresa manipula diversos dados relativos a cada um dos funcionários que são de tipos diferentes.

  – Para cada funcionário deve-se ter sua matrícula, nome, endereço, cargo, o número de dependentes, salário, data de admissão, etc.

DEFINIÇÃO

- Foram vistas duas categorias de tipos de variáveis:
  - Tipos básicos ou primitivos
    - char, int, float, double e void
  - Tipo composto homogêneos
    - Arrays
      - Unidimensional
      - Bidimensional

DEFINIÇÃO

- Uma estrutura (struct) ou registro em C é uma coleção de um ou mais valores agrupados sob um único nome.

- São utilizadas para agrupar informações relacionadas de tipos de dados diferentes.

- Tem à possibilidade de tratar um grupo de valores como uma única variável.

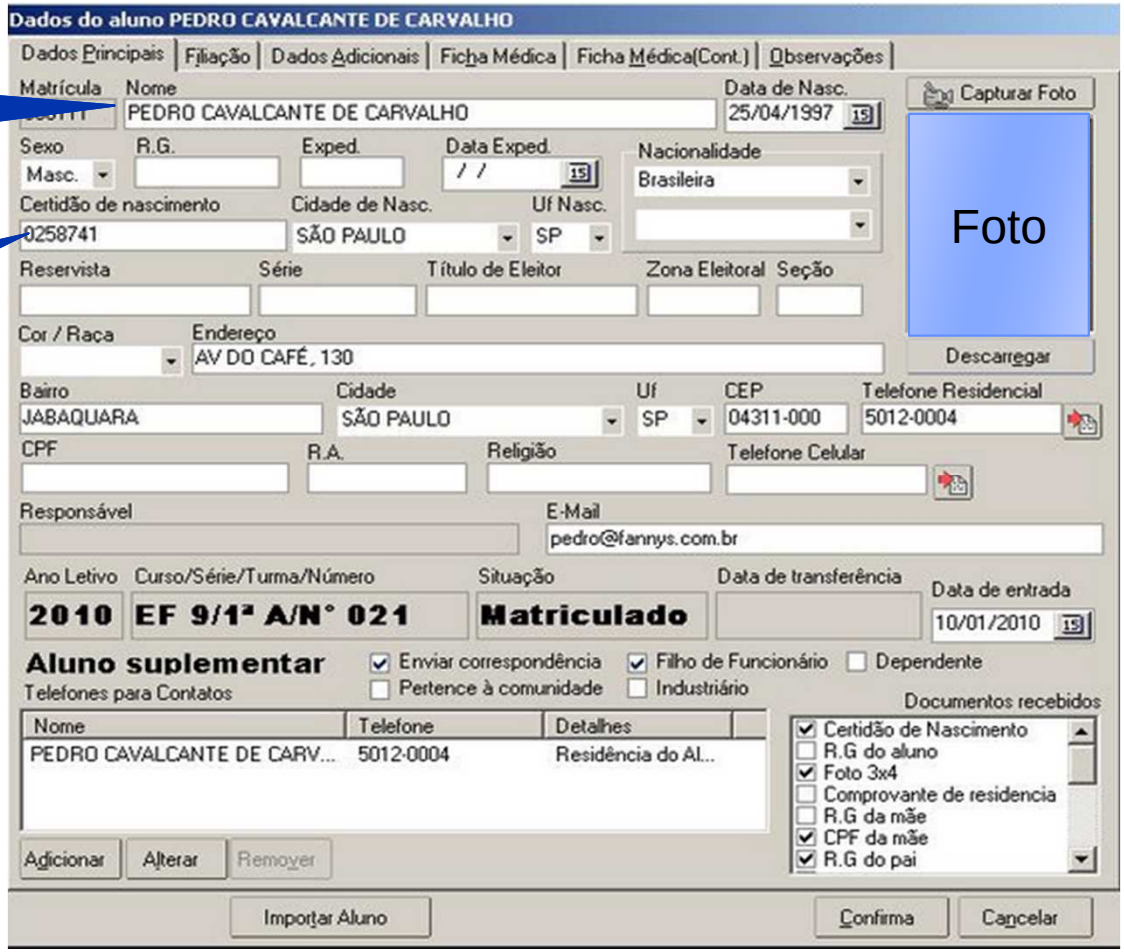
Estrutura de Dados e Algoritmos I

Registros (Structs)

INTRODUÇÃO

caracteres

inteiro



DEFINIÇÃO

- A ideia básica por trás da estrutura é criar apenas um tipo de dado que contenha vários membros, que são outras variáveis.

- Em resumo, é a criação de uma variável que contém dentro de si outras variáveis.

Estrutura de Dados e Algoritmos I

Registros (Structs)

DECLARAÇÃO

```
struct nome_do_tipo_da_estrutura{
        tipo_1 nome_1;
        tipo_2 nome_2;
        ...
        tipo_n nome_n;
};
```

- Onde:
  - struct : define que será declarada uma estrutura
  - nome_da_estrutura : nome dado a estrutura
  - tipo_1 : tipo de dado da variável declarada

DECLARAÇÃO

- As estruturas podem ser declaradas em qualquer escopo do programa (global ou local).

- Por se tratar de um novo tipo de dado, muitas vezes é interessante que todo o programa tenha acesso à estrutura.

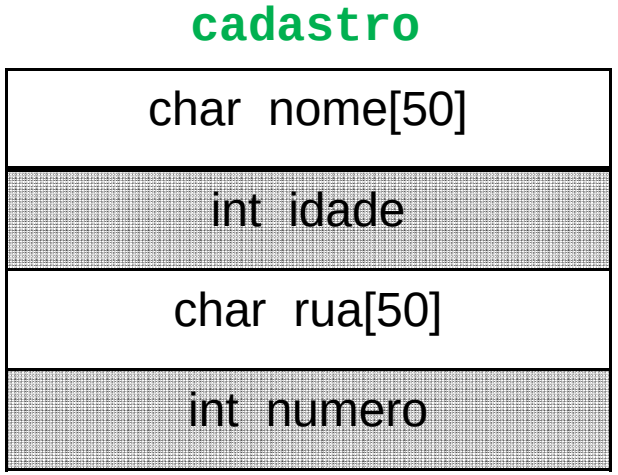
- Daí a necessidade de usar o escopo global.

Estrutura de Dados e Algoritmos I

Registros (Structs)

DECLARAÇÃO

```
struct cadastro{
        char nome[50];
        int idade;
        char rua[50];
        int numero;
};
```



Depois do símbolo de fechar as chaves (}) da estrutura, é necessário colocar um ponto e vírgula.

- Note que os campos da estrutura são definidos da mesma forma que variáveis.

DECLARAÇÃO

- A necessidade do ponto e vírgula é porque pode-se declarar variáveis na definição da estrutura.
- Para isso basta colocar os nomes das variáveis após o comando de fechar as chaves, antes do ponto e vígula.

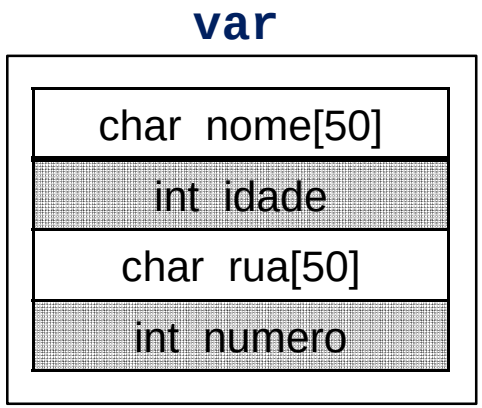
```
struct cadastro{
        char nome[50];
        int idade;
        char rua[50];
        int numero;
} cad1, cad2;
```

DECLARAÇÃO

- Uma vez definida a estrutura, uma variável pode ser declarada de modo similar aos tipos já existentes.

```
struct cadastro var;
```

- Por ser um tipo definido pelo programador, usa-se a palavra struct antes do tipo da nova variável declarada.



DECLARAÇÃO

- O uso de estruturas facilita muito a vida do programador na manipulação dos dados do programa.
- Imagine ter de declarar quatro cadastros para quatro pessoas diferentes:

```
char nome1[50], nome2[50], nome3[50], nome4[50];
int idade1, idade2, idade3, idade4;
char rua1[50], rua2[50], rua3[50], rua4[50];
char numero1, numero2, numero3, numero4;
```

- Utilizando struct, o mesmo pode ser feito da seguinte maneira:

```
struct cadastro c1, c2, c3, c4;
```

ou

```
struct cadastro c[4];
```

ACESSANDO OS CAMPOS

- Cada campo (variável) da estrutura pode ser acessada usando o operador "." (ponto).

```
struct cadastro{
        char nome[50];
        int idade;
        char rua[50];
        int numero;
} Pessoa;

int main( ){
  printf("Nome: ");
  gets(Pessoa.nome);
  printf("Idade");
  scanf("%d", &Pessoa.idade);
  return 0;  }
```

Exemplo de Leitura dos Dados

ACESSANDO OS CAMPOS

- Cada campo (variável) da estrutura pode ser acessada usando o operador "." (ponto).

```
struct cadastro{
        char nome[50];
        int idade;
        char rua[50];
        int numero;
} Pessoa;

int main( ){
  gets(Pessoa.nome);
  printf("Nome escrito %s",Pessoa.nome);
  scanf("%d", &Pessoa.idade);
  printf("Sua idade %d",Pessoa.idade);
  return 0;  }
```

Exemplo de Escrita dos Dados

ATRIBUIÇÃO

- Da mesma forma que as variáveis, sendo que deve referenciar o nome da estrutura declarada.

```
struct cadastro{
        char nome[50];
        int idade;
        char rua[50];
        int numero;
} Pessoa;

int main( ){
  Pessoa.idade = 22;
  Pessoa.numero = 41;
  strcpy(Pessoa.nome, “José”);
  strcpy(Pessoa.rua, “Av. Costa”);
  return 0;  }
```

Observe a diferença entre os Tipos de Dados

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo 1

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct funcionario {
   char nome[30];
   int idade;
   char sexo;
   float altura;
};

int main() {
   struct  funcionario  f;

   strcpy(f.nome,"Joao");

   f.idade = 38;
   f.sexo='M';
   f.altura = 1.76;

   printf("Nome: %s\n", f.nome);
   printf("Idade: %d\n", f.idade);
   printf("Sexo: %c\n", f.sexo);
   printf("Altura: %4.2f\n", f.altura);

   getch();
   return 0;
}
```

Exemplo 2

```
#include<stdio.h>
#include<conio.h>

struct ficha {
    int cod;
   char nome[40];
   char telefone[10];
};

int main() {
   struct  ficha aluno, func;

printf("Código do aluno: ");
scanf("%d", &aluno.cod);
printf("Nome do aluno: ");
scanf("%s", &aluno.nome);
printf("Telefone do aluno: ");
scanf("%s", &aluno.telefone);
printf("Código do funcionário: ");
scanf("%d", &func.cod);
printf("Nome do funcionário: ");
scanf("%s", &func.nome);
printf("Telefone do funcionário: ");
scanf("%s", &func.telefone);

if(aluno.cod == func.cod)
    printf("Aluno e Funcíonários são a mesma pessoa!");
else
    printf("Aluno e Funcíonários são pessoas diferentes!");

getch();
return 0;
}
```

Estrutura de Dados e Algoritmos I

Registros (Structs)

Exemplo 3

```
#include<stdio.h>
#include<stdlib.h>

struct produto {
    int codigo;
   char nome[50];
   int qtd;
   float vlrcomp, vlrvend;
} item;

int main() {

printf("** Recebendo os Dados **\n");
printf("\nCodigo. . . : ");
scanf("%d", &item.codigo);

printf("\nNome. . . : ");
fflush(stdin);
gets(item.nome);
printf("\nQuantidade. . . : ");
scanf("%d", &item.qtd);
printf("\nValor da Compra R$. . . : ");
scanf("%f", &item.vlrcomp);
printf("\nValor da Venda R$. . . : ");
scanf("%f", &item.vlrvend);

system("cls");

printf("** Apresentando os Dados **\n");
printf("** Codigo.....: %d\n", item.codigo);
printf("** Nome.....: %s\n", item.nome);
printf("** Quantidade.....: %d\n", item.qtd);
printf("** Valor da Compra.....: %.2f\n", item.vlrcomp);
printf("** Valor da Venda.....: %.2f\n\n", item.vlrvend);
printf("** Lucro Obtido.....: %.2f\n", item.vlrvend –
item.vlrcomp);


system("pause");
return 0;
}
```

Exemplo 4 – Com Arrays

```
#include<stdio.h>
#include<conio.h>

struct produto {
    int codigo;
   char nome[50];
   int qtd;
   float vlrcp, vlrvd;
} item[3];

int main() {

printf("** Recebendo os Dados **\n");

for(int i=0; i < 3; i++) {
      printf("\nCodigo. . . : ");
      scanf("%d", &item[ i ].codigo);

      printf("\nNome. . . : ");
      fflush(stdin);
      gets(item[ i ].nome);
      printf("\nQuantidade. . . : ");
      scanf("%d", &item[ i ].qtd);
      printf("\nValor da Compra R$. . . : ");
      scanf("%f", &item[ i ].vlrcp);
      printf("\nValor da Venda R$. . . : ");
      scanf("%f", &item[ i ].vlrvd);
fflush(stdin);  }

printf("** Apresentando os Dados **\n");

for(int i=0; i < 3; i++) {
printf("** Codigo.....: %d\n\n", item[ i ].codigo);
printf("** Nome.....: %s\n\n", item[ i ].nome);
printf("** Quantidade.....: %d\n\n", item[ i ].qtd);
printf("** Valor da Compra.....: %.2f\n\n", item[ i ].vlrcp);
printf("** Valor da Venda.....: %.2f\n\n", item[ i ].vlrvd);
printf("** Lucro Obtido.....: %.2f\n\n", item[ i ].vlrvd – item[ i ]. vlrcp);
}
getch();
return 0;    }
```

Exemplo 5 – Com Arrays

```
#include<stdio.h>
#include<conio.h>
#define MAX 5

struct  contato{
    char  nome[30], tel[40];
    int idade;
};

int main( ) {
   struct  contato vet[MAX];
   int  i;

//Lendo vetor de registros
for( i=0; i < MAX; i++) {
       printf("Nome: ");
       gets(vet[ i ].nome);
       printf("Telefone: ");
       gets(vet[ i ].tel);
       printf("Idade: ");
       scanf("%d", &vet[ i ].idade);
       fflush(stdin);
}
//Imprimindo vetor de registros
for(i=0; i < MAX; i++) {
    printf("\n\nNome: %s\n", vet[i].nome);
    printf("\nTelefone: %s\n", vet[i].tel);
    printf("\nIdade: %d\n", vet[i].idade);
}
getch();
return 0;  }
```

Exercício

Criar uma estrutura representando um aluno de uma disciplina. Essa estrutura deve conter o número de matrícula do aluno, seu nome e as notas de três provas. Agora, escreva um programa que leia os dados de cinco alunos e os armazene nessa estrutura. Em seguida, exibir o nome e as notas do aluno que possui a maior média geral dentre os cinco alunos:

ESTRUTURA DE DADOS E ALGORITMOS I

Estrutura de Registros

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
2º. Período