ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

Estruturas: typedef

- Todas as vezes em que se for declarar uma variável do tipo estrutura fora do local de definição da mesma, é necessário colocar antes do nome da variável a palavra reservada struct seguida do nome da estrutura.

Ex: struct Data d1; => declaração da variável d1

- Existe uma outra maneira de se declarar uma variável do tipo estrutura, utilizando uma palavra sinônimo para a estrutura. Isto é feito através da palavra reservada typedef, cuja sintaxe é: typedef tipo_existente sinônimo .

Estruturas: typedef

- A palavra typedef não cria um novo tipo, ela apenas permite que um determinado tipo possa ser denominado de forma diferente.

```
struct Aluno{
    int codigo;
    char nome[200];
    float nota1;
    float nota2;
        };
```

```
typedef struct Aluno Dt;
Dt d1;
```

```
struct Aluno aluno_especial;
```

Neste exemplo foi criado o sinônimo Dt para o tipo struct Aluno. Deste modo, a declaração da variável d1 foi feita utilizando esse sinônimo.

Estruturas: typedef

- A palavra typedef não cria um novo tipo, ela apenas permite que um determinado tipo possa ser denominado de forma diferente.

```
struct Aluno{
    int codigo;
    char nome[200];
    float nota1;
    float nota2;
        };
```

```
typedef struct Aluno Dt;
Dt d1;
```

```
struct Aluno aluno_especial;
Dt aluno_especial;
```

Neste exemplo foi criado o sinônimo Dt para o tipo struct Aluno. Deste modo, a declaração da variável d1 foi feita utilizando esse sinônimo.

Estruturas: typedef

- A utilização da palavra reservada typedef não se limita apenas as estruturas, ela estabelece um sinônimo para qualquer conjunto de palavras, por ex:

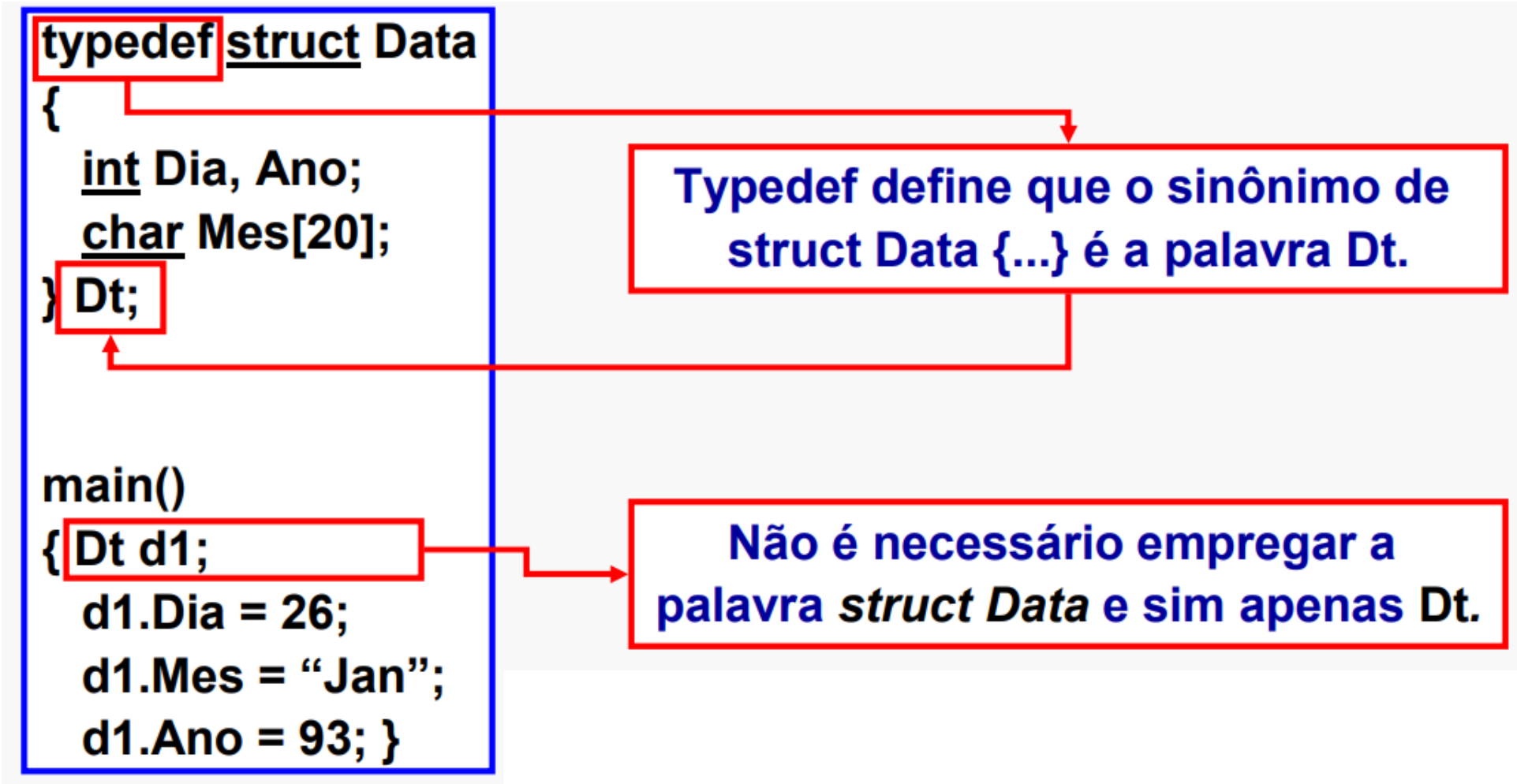
- typedef float real;

- O tipo float passa também a ser representado pela palavra real. Pode-se então declarar uma variável utilizando as duas palavras: float e real

```
float x;
real y;
```

Estruturas: typedef

- Pode-se também empregar a palavra reservada typedef junto com a definição da estrutura:



Estruturas: typedef

```
struct Data
{
   int Dia, Ano;
   char Mes[20];
};

main()
{  struct Data d1;
   d1.Dia = 26;
   d1.Mes = "Jan";
   d1.Ano = 93;}
```

```
struct Data
{
   int Dia, Ano;
   char Mes[20];
};

typedef struct Data Dt;
main()
{  Dt d1;
   d1.Dia = 26;
   d1.Mes = "Jan";
   d1.Ano = 93; }
```

```
typedef struct Data
{
   int Dia, Ano;
   char Mes[20];
} Dt;


main()
{ Dt d1;
   d1.Dia = 26;
   d1.Mes = "Jan";
   d1.Ano = 93; }
```

Funções com Registros

- A ideia de usar uma **struct** é permitir que, ao armazenar os dados de uma mesma entidade, isto possa ser feito com uma única variável. Por exemplo, se for preciso armazenar a *altura*, o *peso* e a idade de uma pessoa, pode-se criar uma **struct** chamada **Pessoa** e agrupar os dados em um único tipo de dado.

```
struct Pessoa// Cria uma STRUCT para armazenar os dados de uma pessoa
{
    float Peso;   // define o campo Peso
    int Idade;    // define o campo Idade
    float Altura; // define o campo Altura
};
```

Funções com Registros

Após a criação do tipo, é possível declarar variáveis do tipo Pessoa, desta forma:
Pessoa Joao, P1, P2;
Pessoa Povo[10]; // cria um vetor de 10 pessoas.

Para acessar os campos de uma struct, usa-se a sintaxe NomeDaVariavel.NomeDoCampo

```
struct Pessoa//
{
   float Peso;   // define o campo Peso
   int Idade;    // define o campo Idade
   float Altura; // define o campo Altura
};
```

```
Joao.Idade = 15;
Joao.Peso = 60.5;
Joao.Altura = 1.75;

Povo[4].Idade = 23;
Povo[4].Peso = 75.3;
Povo[4].Altura = 1.89;
```

```
P2 = Povo[4];
```

Funções com Registros

Passagem de Structs por Parâmetro

- Para passar uma struct por valor basta declará-la como um dos parâmetros

```
void ImprimePessoa(Pessoa P) // declara o parâmetro como uma struct
{
  printf("Idade: %d  Peso: %f Altura: %f\n", P.Idade, P.Peso, P.Altura);
}
```

```
struct Pessoa//
{
    float Peso;    // define o campo Peso
    int Idade;     // define o campo Idade
    float Altura; // define o campo Altura
};
```

Funções com Registros

Passagem de Structs por Parâmetro

- Para passar uma struct por valor basta declará-la como um dos parâmetros

```
void ImprimePessoa(Pessoa P) // declara o parâmetro como uma struct
{
  printf("Idade: %d  Peso: %f Altura: %f\n", P.Idade, P.Peso, P.Altura);
}
```

```
// chama a função que recebe a struct como parâmetro
    ImprimePessoa(Joao);
    ImprimePessoa(Povo[4]);
    ImprimePessoa(P2);
```

```
struct Pessoa//
{
    float Peso;   // define o campo Peso
    int Idade;    // define o campo Idade
    float Altura; // define o campo Altura
};
```

Funções com Registros

```
void ImprimePessoa(Pessoa P) // declara o parâmetro como uma struct
{
  printf("Idade: %d  Peso: %f Altura: %f\n", P.Idade, P.Peso, P.Altura);
}
```

```
struct Pessoa//
{
   float Peso;   // define o campo Peso
   int Idade;    // define o campo Idade
   float Altura; // define o campo Altura
};
```

```
Pessoa SetPessoa(int idade, float peso, float altura)
{
    Pessoa P;
    P.Idade = idade;
    P.Peso = peso;
    P.Altura = altura;
    return P; // retorna a struct contendo os dados passados por parâmetro
 }
```

```
Int main ()
{
    Pessoa Joao;

    Joao = SetPessoa(15,60.5,1.75);
    ImprimePessoa(Joao);
   return 0;
}
```

Registros

Escrever um programa que cadastre produtos. Para cada produto devem conter seguintes dados: nome, código, preço e quantidade do produto.

• Adicione 5 produtos à loja.

• Imprima uma lista com o código e nome de cada produto.

• Leia um valor x e mostre as informações de todos os produtos com o preço menor que x.

FIM!