**STRUCTS**

**Structs**, também conhecidas como **Registros**, definem tipos de dados que agrupam variáveis sob um mesmo tipo de dado.

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.

typedef struct // 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  
} Pessoa; // Define o **nome** do novo tipo criado

Após a criação do tipo, é possivel 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, conforme o exemplo a seguir.

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;

Outra vantagem de utilizar **struct** é a possibilidade de atribuir os dados de uma **struct** para outra, com apenas um comando de atribuição, como neste exemplo:  
  
P2 = Povo[4];

**Passagem de Structs por Parâmetro**

Para passar uma struct **por valor** basta declará-la como um dos parâmetros, como no exemplo a seguir  
  
#include <stdio.h>  
typedef struct // 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  
} Pessoa; **// Define o nome do novo tipo criado**

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

int main()  
{  
    Pessoa Joao, P2;   
    Pessoa Povo[10];   
  
    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];  
          P2.Idade++;

// chama a função que recebe a struct como parâmetro  
    ImprimePessoa(Joao);  
    ImprimePessoa(Povo[4]);  
    ImprimePessoa(P2);  
   return 0;  
}

**Retorno de Structs em Funções**  
  
Como uma struct define um tipo de dado, este tipo pode ser retornado em uma função, da mesma forma que ocorre com qualquer outro tipo de dado.  
No exemplo a seguir, a função retorna uma struct que conterá, em seus campos, os dados que foram recebidos por parâmetro.  
  
#include <stdio.h>

typedef struct // 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  
} Pessoa; // Define o **nome** do novo tipo criado  
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  
}  
void ImprimePessoa(Pessoa P) // declara o parâmetro como uma struct   
{  
  printf("Idade: %d  Peso: %f Altura: %f\n", P.Idade, P.Peso, P.Altura);   
}

int main()  
{  
    Pessoa Joao;  
    Joao = SetPessoa(15,60.5,1.75); // atribui o retorno da função a uma variável struct  
    ImprimePessoa(Joao);

return 0;  
 }

**Passagem de Structs por Referência**Para passar uma struct por referência,deve-se passar um ponteiro para a struct, como no exemplo a seguir.  
  
#include <stdio.h>  
  
typedef struct // 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  
} Pessoa; // Define o nome do novo tipo criado  
  
void ImprimePessoa(Pessoa P)  
{  
  printf("Idade: %d  Peso: %f Altura: %f\n", P.Idade, P.Peso, P.Altura);  
}  
  
void SetPessoa(Pessoa \*P, int idade, float peso, float altura)   
{ // Nesta função o parâmetro P é um ponteiro para uma struct   
    (\*P).Idade = idade;  // o campo pode ser acessado desta forma   
    P->Peso = peso;      // ou desta  
    P->Altura = altura;  
}  
      
int main()  
{  
    Pessoa Joao;  
    SetPessoa(&Joao, 15, 70.5, 1.75);  
    ImprimePessoa(Joao);  
  
    return 0;  
}