Rodrigo Henrich

rodrigohenrich@faccat.br

Variáveis compostas em C

- Em C é possível definir tipos especiais de variáveis;
- Este tipo é basicamente uma variável com outras variáveis dentro
- Pensando em uma pessoa, podemos pensar nos seguintes dados sobre ela
  - nome;
  - idade;
  - gênero;

Variáveis compostas em C

- Para criar uma struct em C vamos usar a seguinte sintaxe

  struct <identificador>{

  <listagem dos tipos e membros>;

  }

Variáveis compostas em C

- Para usar o tipo recém definido

  struct <identificador> <variável>;

Exemplo de uso da criação de uma struct

```c
#include<stdio.h>

struct pessoa{
   char nome[50];
   int idade;
   char genero;
};

int main(){
   struct pessoa p1;
}
```

- nossa struct chama **pessoa**
- o **nome** é um char de até 50 caracteres
- a **idade** é um inteiro
- o **gênero** é um char (M ou F)
- dentro do main() para criar uma variável usando nossa struct
- struct pessoa p1
- **p1** é o nome da nossa variável

Exemplo de uso da criação de uma struct

```c
#include<stdio.h>

struct pessoa{
    char nome[50];
    int idade;
    char genero;
} p1;

int main(){
    //struct pessoa p1;
}
```

- neste exemplo já declaramos a variável p1 logo na sequência da declaração da struct;

Exemplo 1

Crie uma struct que represente um aluno, ela deve conter

- nome
- matrícula
- três notas trimestrais
- número de faltas
- status (A - aprovado ou R - reprovado)

Acessando os membros da struct

- Para acessar os membros de uma struct usamos o ‘.’
- Com a seguinte sintaxe
- No nosso primeiro supondo a struct abaixo:

```
3  struct pessoa{
4      char nome[50];
5      int idade;
6      char genero;
7  } p1;
```

Acessando os membros da struct

- Para acessar (ler ou escrever) um valor no atributo
- Tenho que acessar
- **p1.idade**
- **p1.genero**
- **p1.nome**

```
3  struct pessoa{
4      char nome[50];
5      int idade;
6      char genero;
7  } p1;
```

Acessando os membros da struct

```
10  int main(){
11      setlocale(LC_ALL,"");
12      struct pessoa p1;
13      printf("Entre com seu nome: ");
14      scanf("%s",p1.nome);
15      setbuf(stdin,NULL);
16      printf("Entre com sua idade ");
17      scanf("%d",&p1.idade);
18      setbuf(stdin,NULL);
19      printf("Selecione um gênero M/F ");
20      scanf("%c",&p1.genero);
21      printf("----------------------\n");
22      printf("Nome: %s\n", p1.nome);
23      printf("Idade: %d\n",p1.idade);
24      printf("Gênero: %c\n",p1.genero);
25  }
```

Exemplo 2

Leia do teclado o nome, as notas e a frequência de dois alunos, armazene na estrutura criada.

Em seguida calcule a média e verifique se o aluno está aprovado ou reprovado e altere seu status.

Para ser aprovado ele deve ter 75% de frequência, de um total de 40 aulas e pelo menos 8 de média

inicializando uma variável struct

- Podemos também definir os dados de uma struct logo na criação dela
- Para isso é necessário informar todos os valores que ela pede no formato de conjunto, cada valor separado por ,
- Ex.: { valor1, “valor2”, valorn}
- No nosso exemplo para criar uma pessoa e já inicializar os valores para ela usamos

inicializando uma variável struct

```
10  int main(){
11      setlocale(LC_ALL,"");
12      struct pessoa p1;
13      struct pessoa p2 = {"Ana Julia", 23, 'F'};
14      printf("----------------------\n");
15      printf("Nome: %s\n", p2.nome);
16      printf("Idade: %d\n",p2.idade);
17      printf("Gênero: %c\n",p2.genero);
18  }
```

Exemplo

Modifique o exercício do exemplo 2 para

Já ter os nomes, às notas e a frequência dos alunos na criação deles

Logo depois calcular a média e verificar o status do aluno

Atualizar as informações na struct e mostrar na tela

typedef

- A função typedef, permite definir um novo nome para tipos de dados já existentes na linguagem
- Sua sintaxe é **typedef** <nome do tipo de dado> <novo nome>;
- Por exemplo, queremos criar o tipo de dado nota, que na verdade é um float
- Para isso podemos fazer o seguinte

```
typedef float  nota;
```

typedef

```c
#include<stdio.h>
#include<locale.h>

typedef float  nota;

int main(){
   nota avaliacaoUm, avaliacaoDois, media;
   avaliacaoUm = 6;
   avaliacaoDois = 8;
   media = (avaliacaoUm+avaliacaoDois)/2;
   printf("%.2f",media);
}
```

typedef

- Voltando as structs agora é possível definir um tipo de dados usando nossas structs
- Dessa forma podemos criar vetores deste novo tipo
- Usá-lo como parâmetro e tipo de retorno em nossas funções, sem precisar usar a palavra struct a cada nova variável declarada.
- Para definir a nossa struct como um novo tipo de dado em nosso programa, vamos usar uma mistura de **typedef** com **struct**

typedef

- Agora o nome para a struct é opcional, pois não vamos precisar dele
- E no final depois de definir a struct temos que passar um nome para nosso novo tipo de dado
- no exemplo **typedef struct**{...} **Pessoa**;
- O uso da nossa struct continua muito parecido

```
4  typedef struct {
5      char nome[50];
6      int idade;
7      char genero;
8  } Pessoa;
```

typedef

```c
#include<stdio.h>
#include<locale.h>

typedef struct {
   char nome[50];
   int idade;
   char genero;
} Pessoa;

int main(){
   setlocale(LC_ALL,"");
   Pessoa p1 = {"Ana Julia", 23, 'F'};
   printf("----------------------\n");
   printf("Nome: %s\n", p1.nome);
   printf("Idade: %d\n",p1.idade);
   printf("Gênero: %c\n",p1.genero);
}
```

vetores de structs

- Assim como outros tipos
- podemos criar vetores do nosso novo tipo de dado, ou seja vetores de structs
- A sintaxe de criação é exatamente a mesma de vetores de outras variáveis

vetores de structs

- Por exemplo se eu precisar de uma lista de 3 pessoas
- A sintaxe da criação desta lista seria a seguinte

```
Pessoa listaPessoa[3];
```

vetores de structs

- Para acessar os membros desta lista agora é preciso indicar o índice do elemento que desejo acessar
- Para isso:
  - listaPessoa[0].nome;
  - listapessoa[0].idade;
  - listaPessoa[0].genero

vetores de structs

```
10  int main(){
11    setlocale(LC_ALL,"");
12    Pessoa listaPessoa[3];
13    for(int i=0;i<3;i++){
14      printf("Entre com o nome da pessoa %d ",i+1);
15      scanf("%s",listaPessoa[i].nome);
16      setbuf(stdin,NULL);
17      printf("Entre com a idade do %s ",listaPessoa[i].nome);
18      scanf("%d",listaPessoa[i].idade);
19      setbuf(stdin,NULL);
20      printf("Selecione o gênero (M/F) do %s ",listaPessoa[i].nome);
21      scanf("%c",&listaPessoa[i].genero);
22    }
23  }
```

vetores de structs

- Modifique o programa anterior para que após a leitura liste as pessoas na tela.

structs em funções

- Criando uma função para ler as informações de uma pessoa do teclado

```
10  Pessoa lePessoa(){
11      Pessoa p;
12      printf("Entre com seu nome: ");
13      scanf("%s",p.nome);
14      setbuf(stdin,NULL);
15      printf("Entre com sua idade ");
16      scanf("%d",&p.idade);
17      setbuf(stdin,NULL);
18      printf("Selecione um gênero M/F ");
19      scanf("%c",&p.genero);
20      return p;
21  }
```

structs em funções

- Criando uma função para mostrar as informações de uma pessoa na tela

```
22  //Função para mostrar os dados de uma pessoa na tela
23  void mostraPessoa(Pessoa p){
24      printf("Nome: %s\n", p.nome);
25      printf("Idade: %d\n",p.idade);
26      printf("Gênero: %c\n",p.genero);
27  }
```

structs em funções

- Usando as funções no meu programa...

```
29  int main(){
30      setlocale(LC_ALL,"");
31      Pessoa pessoa1;
32      //Usando nossa função para obter dos dados de uma pessoa
33      pessoa1 = lePessoa();
34      //Usando a função que mostra os dados da pessoa na tela
35      mostraPessoa(pessoa1);
36  }
```

Exemplo

- Modifique o programa do exemplo 3 para
- Existir o tipo de dado Aluno
- Crie funções para ler as informações de um aluno
- Crie outra função para determinar de ele está aprovado ou não
- Crie outra função para mostrar os dados do aluno na tela