

Redefinição de tipos, Registros, Enumeração

Prof. Tiago A. Almeida
talmeida@ufscar.br

PARTE 1

Redefinição de tipos

Prof. Tiago A. Almeida
talmeida@ufscar.br

- ✓ Às vezes, **por questão de organização**, gostaríamos de **criar um tipo próprio nosso**, que faz exatamente a mesma coisa que um outro tipo já existente.
- ✓ Isso é útil quando desenvolvemos programas grandes nos quais a alteração do tipo de uma determinada variável para outra acarretaria na alteração de muitas variáveis.
- ✓ Por exemplo, em um programa onde manipulamos médias de alunos, todas as variáveis que trabalhassem com nota poderiam ser do tipo `nota`, e não `int` ou `float`.

- ✓ A forma de se fazer isso é utilizando o comando `typedef`, e seguindo a estrutura abaixo:

```
typedef <tipo já existente> <tipo novo>;
```

- ✓ Usualmente, fazemos essa declaração fora da função `main()`, embora seja permitido fazer dentro.

- ✓ Ex: `typedef float nota;`

Cria um novo tipo, chamado `nota`, cujas variáveis desse tipo serão pontos flutuantes.

```
#include <stdio.h>
typedef float nota;
main () {
    nota P1;
    printf ("Digite a nota 1\n");
    scanf ("%f", &P1);
    printf ("A nota 1 foi %f\n", P1);
}
```

Veja mais detalhes em `typedef.c`

Exemplo de uso do typedef

```
#include <stdio.h>
typedef float nota;
main () {
    nota P1;
    printf ("Digite a nota 1\n");
    scanf ("%f", &P1);
    printf ("A nota 1 foi %f\n", P1);
}
```

Veja mais detalhes em `typedef.c`

Exemplo de uso do typedef

```
#include <stdio.h>
typedef float nota;
main () {
    nota P1;
    printf ("Digite a nota 1\n");
    scanf ("%f", &P1);
    printf ("A nota 1 foi %f\n", P1);
}
```

Veja mais detalhes em `typedef.c`

Exemplos de uso do typedef

```
...

typedef int    Boolean;
typedef float Dollar;
typedef int    Iterator;
typedef unsigned long int Quantity;

...

Boolean    flag;
Dollar    cash_in, cash_out;
Iterator   i, j;
Quantity   qt_books, qt_magazines;

...
```


PARTE 2

Registros (*Structs*)

Prof. Tiago A. Almeida
talmeida@ufscar.br

✓ Um registro é uma variável que contém diversas variáveis (chamadas campos), usualmente de tipos diferentes, mas que dentro de um determinado contexto fazem sentido se agrupadas. Podemos comparar um registro com uma ficha que possui diversos dados sobre uma determinada entidade.

✓ Exemplos:

- * Registro de alunos (campos: nome, RA, médias de provas, médias de trabalhos, SAC, etc...)
- * Registro de pacientes (campos: nome, endereço, histórico de doenças, etc...)

1. Family Name			
2. First (Given) Name		3. Birth Date (DD/MM/YY)	
4. Country of Citizenship		5. Sex (Male or Female)	
6. Passport Issue Date (DD/MM/YY)		7. Passport Expiration Date (DD/MM/YY)	
8. Passport Number		9. Airline and Flight Number	
10. Country Where You Live		11. Country Where You Boarded	
12. City Where Visa Was Issued		13. Date Issued (DD/MM/YY)	
14. Address While in the United States (Number and Street)			
15. City and State			
16. Telephone Number in the U.S. Where You Can be Reached			
17. Email Address			

1. Family Name	
2. First (Given) Name	3. Birth Date (DD/MM/YY)
4. Country of Citizenship	5. Sex (Male or Female)
6. Passport Issue Date (DD/MM/YY)	7. Passport Expiration Date (DD/MM/YY)
8. Passport Number	9. Airline and Flight Number
10. Country Where You Live	11. Country Where You Boarded
12. City Where Visa Was Issued	13. Date Issued (DD/MM/YY)
14. Address While in the United States (Number and Street)	
15. City and State	
16. Telephone Number in the U.S. Where You Can be Reached	
17. Email Address	

CBP Form I-94 (05/08)

Registro

**Campos
do
registro**

1. Family Name			
2. First (Given) Name		3. Birth Date (DD/MM/YY)	
4. Country of Citizenship		5. Sex (Male or Female)	
6. Passport Issue Date (DD/MM/YY)		7. Passport Expiration Date (DD/MM/YY)	
8. Passport Number		9. Airline and Flight Number	
10. Country Where You Live		11. Country Where You Boarded	
12. City Where Visa Was Issued		13. Date Issued (DD/MM/YY)	
14. Address While in the United States (Number and Street)			
15. City and State			
16. Telephone Number in the U.S. Where You Can be Reached			
17. Email Address			

CBP Form I-94 (05/08)

Declarando o formato do registro

✓ A primeira parte da criação de um registro é declarar seu formato.

* Isso é feito utilizando a palavra-chave **struct**

```
struct nome_do_tipo_do_registro {  
    tipo_1 nome_campo_1;  
    tipo_2 nome_campo_2;  
    tipo_3 nome_campo_3;  
    ...  
    tipo_n nome_campo_n;  
};
```

- ✓ A declaração do formato de uma estrutura pode ser feita dentro do seu programa (ou seja, na função **main**) ou fora dela. Usualmente, ela é feita fora da função, como no exemplo abaixo:

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

```
/* Declare o formato de seu registro aqui */
```

```
int main() {
```

```
/* Construa seu programa aqui */
```

```
}
```

Declarando o formato do registro

- ✓ A primeira etapa é declarar uma variável do tipo **struct nome_do_tipo_da_estrutura**, que será usada dentro de seu programa, como no exemplo:

```
#include <stdio.h>

struct ficha {
    int ra;
    float media;
};

int main() {
    struct ficha f;

    return (0);
}
```


Utilizando campos de um registro

- ✓ Podemos acessar individualmente os campos de um determinado registro como se fossem variáveis normais, utilizando a seguinte estrutura:

`nome_do_registro.nome_do_campo`

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

```
struct ficha {
    int ra;
    float media;
};
```

```
int main() {
    struct ficha f;

    return (0);
}
```

✓ Para o registro declarado anteriormente, utilizaríamos

f.ra

para acessar o campo `ra` do registro `f`

✓ Podemos colocar o campo de um registro em qualquer lugar onde colocaríamos uma variável.

✓ A leitura dos campos de um registro a partir do teclado deve ser feita campo a campo, como se fossem variáveis independentes.

```
printf ("Digite o ra do aluno: ");  
scanf ("%d", &f.ra);
```

```
printf ("Digite a média do aluno: ");  
scanf ("%f", &f.media);
```

✓ A escrita na tela do valor dos campos de um registro **deve ser feita campo a campo**, como se fossem variáveis independentes.

```
printf ("O aluno %d tirou média %f\n",  
        f.ra, f.media);
```

Veja o exemplo em `structs.c`

- ✓ A inicialização pode ser feita de **duas formas**: **na ordem** de aparecimento dos campos (sem necessidade de informar o nome do campo) ou **em qualquer ordem** (precisa informar o nome do campo)

```
struct aluno aluno1 = {987654, 5.5, 4.9, 5.2},  
    aluno2 = {.ra = 987654, .nota_p1 = 5.5,  
             .nota_p2 = 4.9};
```

Veja o exemplo em `structs_inicializacao.c`

- ✓ A cópia de um registro pode ser feita como se fosse a cópia de uma variável normal, ou seja

```
registro_1 = registro_2
```

Veja o exemplo em `structs_copia.c`

✓ É possível redefinir um registro para nomeá-lo de acordo com a semântica do que ele representa

* Forma 1:

```
struct aluno {  
    int    ra;  
    nota   nota_p1, nota_p2;  
    float  media;  
};  
  
typedef struct aluno Aluno;
```

Veja o exemplo em `structs_typedef.c`

✓ É possível redefinir um registro para nomeá-lo de acordo com a semântica do que ele representa

* Forma 2:

```
typedef struct aluno {  
    int    ra;  
    nota   nota_p1, nota_p2;  
    float  media;  
} Aluno;
```

Veja o exemplo em `structs_typedef.c`

- ✓ Pode-se também declarar um registro como um campo de outro registro (Ver `structs_aninhadas.c`)

```
typedef float nota;  
  
struct _provas {  
    nota p1;  
    nota p2;  
    nota p3;  
} provas;  
  
typedef struct _aluno {  
    int ra;  
    provas notas;  
    nota media;  
} aluno;
```

PARTE 3

Tipos enumerados

Prof. Tiago A. Almeida
talmeida@ufscar.br

- ✓ Para criar uma variável para armazenar um determinado mês do ano, uma das soluções possíveis é criar um inteiro e armazenar um número associado àquele mês. Assim, janeiro seria o mês número 1, fevereiro o mês número 2, e assim sucessivamente.
- ✓ Para efeito de compreensão e organização do programa, seria melhor se pudéssemos escrever no código

`mes = janeiro`

✓ O comando **enum** cria um tipo enumerado, ou seja, um tipo que funciona como um inteiro, mas para o qual estão associadas constantes numeradas que podem ser utilizadas como constantes inteiras.

✓ Sua sintaxe é

```
enum <nome> {<constante1>, <constante2>,  
              ..., <constanteN>};
```

- ✓ O compilador associa o número zero para o primeiro item e associa para o item i o número $i - 1$. Ex:

```
enum mes { Jan, Feb, Mar, Abr, Mai, Jun,  
          Jul, Ago, Set, Out, Nov, Dez };
```

- ✓ Aqui, Janeiro corresponde a 0, Fevereiro a 1 e assim sucessivamente, até dezembro que corresponde ao número 11.

Usando um tipo enumerado

- ✓ Declara-se uma variável do tipo enumerado utilizando o nome do tipo que você escolheu. Ex:

```
enum mes mes_aniversario;
```

- ✓ Você pode usar o tipo enumerado em qualquer lugar em que e você utilizaria um inteiro. Ex:

```
printf ("%d", mes_aniversario) ;
```

Veja o exemplo em `enum.c`

- ✓ Você pode atribuir um valor inicial para qualquer um dos elementos do tipo enumerado, bastando substituir

`<constanten>` por `<constanten> = <Valorn>`.

```
enum mes { Jan=1, Feb, Mar, Abr, Mai, Jun,  
           Jul, Ago, Set, Out, Nov, Dez };
```

- ✓ Aqui, Janeiro corresponde a 1, Fevereiro a 2 e assim sucessivamente, até Dezembro que corresponde ao número 12.

Exercícios

1. Escreva um programa para receber e imprimir na tela uma data no formato DD/MM/AAAA (use `struct`). Teste se a data é válida (se está entre 01/01/2000 e 31/12/2010) e solicite novamente até que o usuário informe uma data válida.
2. Escreva um programa que receba uma data válida e escreva na tela o mês por extenso (use tipo enumerado). Ex: 20/05/2010 => 20 de Maio de 2010.
3. Escreva um programa para manter o cadastro de três alunos na disciplina. A média de cada aluno é computada através da média simples entre duas provas e um trabalho. Primeiro, receba os dados da disciplina e de cada aluno e, em seguida, imprima a média geral da turma. Use `structs` aninhadas e redefinição de tipos.

```
Avaliacoes {p1, p2, trab}
```

```
Aluno {ra, notas, media}
```

```
Disciplina {codigo, qtAlunos, media}
```