

28. Registros, ponteiros, sub-rotinas

DIM0321

2015.1

Outline

- 1 Introdução
- 2 Registros & ponteiros
- 3 Registros & sub-rotinas

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 1 / 18

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 2 / 18

1 Introdução

2 Registros & ponteiros

3 Registros & sub-rotinas

Lembretes

- Registros são apenas um **outro tipo de dados**.
- Ponteiros de registros podem ser declarados e usados como foi o caso para inteiros, pontos flutuantes. . .
- e como será o caso

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 3 / 18

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 4 / 18

1 Introdução

2 Registros & ponteiros

3 Registros & sub-rotinas

Sintaxe e uso

```
1 int main (void)
2 {
3     typedef struct {
4         unsigned int dia;
5         unsigned int mes;
6         unsigned int ano;
7     } data;
8
9     data hoje = { 21, 12, 2014 };
10    data *p_data;
11    data *p_data2 = &hoje;
12
13    p_data = &hoje;
14
15    (*p_data).dia = 6;    /* os parênteses envolvendo *p_data são necessários */
16    (*p_data).mes = 5;    /* devido ao operador ., de seleção de campo de registro, */
17    (*p_data).ano = 2013;
18
19
20    p_data->dia = 6;    /* os parênteses envolvendo *p_data são necessários */
21    p_data->mes = 5;    /* devido ao operador ., de seleção de campo de registro, */
22    p_data->ano = 2013;
23 }
```

Alocação

- Registros são alocados em memória **sequencialmente**.
- Podemos usar os operadores de referenciamento & e dereferenciamento *.

Uso de ->

- Existe um outro operador de acesso indireto aos campos de um registro, >
- Ele permite substituir o uso simultâneo dos operadores * e . no caso dos ponteiros.

Exemplo

```
1 #include <stdio.h>
2 typedef struct {
3     int dia;
4     int mes;
5     int ano;
6 } data;
7
8 int main(void)
9 {
10    data hoje;
11    data *p_data;
12
13    p_data = &hoje;
14    p_data->dia = 6;
15    p_data->mes = 5;
16    p_data->ano = 2013;
17
18    printf("data de hoje: %.2i/%.2i/%.4i\n", hoje.dia, hoje.mes, hoje.ano);
19    return 0;
20 }
```

Ponteiros como campo de registro

Atenção à sintaxe

Exemplo

Em resumo

```
1 #include <stdio.h>
2 typedef struct {
3     int *ptr1;
4     int *ptr2;
5 } registro;
6
7 int main(void)
8 {
9     int i1, i2 = 100;
10    registro reg, *reg_ptr;
11    reg.ptr1 = &i1;
12    reg.ptr2 = &i2;
13    reg_ptr = &reg;
14    *reg.ptr1 = 35;
15    *(*reg_ptr).ptr1 = *reg.ptr1 / 7;
16    *reg_ptr->ptr1 = *reg_ptr->ptr1 + 5;
17    *reg_ptr->ptr2 -= 50;
18
19    printf("%i, %i, %i, %i\n", i1, *reg.ptr1, *(*reg_ptr).ptr1, *reg_ptr->ptr1);
20    printf("%i, %i, %i, %i\n", i2, *reg.ptr2, *(*reg_ptr).ptr2, *reg_ptr->ptr2);
21    return 0;
22 }
```

1 Introdução

2 Registros & ponteiros

3 Registros & sub-rotinas

Uso nas sub-rotinas

- Registros, com 0 tipos compostos heterogêneos, também podem ser passados como parâmetros nas sub-rotinas.
- Diferentemente dos arranjos que são ponteiros e não copiam dados, um registro **copia** todos os seus dados na passagem de parâmetro tal qual ocorre com os tipos básicos.
- Para evitar a cópia dos dados, podemos novamente fazer uso explícito de ponteiros.

Passagem como parâmetro

Modo

- Pode-se passar:
 - um registro inteiro
 - apenas um campo,
 - ou o endereço de um deles como parâmetro de sub-rotina.
- Todas as passagens são **por valor** i.e. cópias.

Retornar um registro

Observação

- Sub-rotinas podem retornar registros, porque eles são estruturas de dados.
- Só arranjo (i.e ponteiros) não precisam ser retornados porque eles podem ser modificados **inplace** (no lugar)

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 13 / 18

Exemplo

```
1 #include <stdio.h>
2
3 typedef struct {
4     int codigo;
5     char descricao[50];
6     float valor;
7     int estoque;
8 } produto;
9
10 void imprime_produto(produto prod)
11 {
12     printf("%s (cod. %i): R$%.2f\n", prod.descricao, prod.codigo, prod.valor);
13 }
14
15 produto altera_preco(produto prod, float novo_valor)
16 {
17     prod.valor = novo_valor;
18     return prod;
19 }
20
21 int main(void)
22 {
23     produto produto = { 104, "caneta", 3.25, 1598 };
24     imprime_produto(produto);
25     produto = altera_preco(produto, 2.19);
26     imprime_produto(produto);
27     return 0;
28 }
```

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 15 / 18

Alinhamento em memória

```
1 #include <stdio.h>
2
3 typedef struct {
4     char c;
5     int i;
6 } data1;
7
8 typedef struct {
9     int i;
10    char c;
11 } data2;
12
13 int main(void)
14 {
15     data1 d1;
16     data2 d2;
17
18     printf("char: %p, int: %p\n", &d1.c, &d1.i);
19     printf("char: %p, int: %p\n", &d2.c, &d2.i);
20     d2.c = 2;
21     printf("%i\n", d2.c);
22     *(&d2.i + 1) = 4;
23     printf("%i\n", d2.c);
24     return 0;
25 }
```

DIM0321

28. Registros, ponteiros, sub-rotinas

2015.1 16 / 18

Referências

Precisões

[KR88] 6

[Bac13] 8.1

-  André Backes, *Linguagem C completa e descomplicada*, Elsevier, 2013.
-  Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

Perguntas ?



<http://dimap.ufrn.br/~richard/dim0321>