

Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Roteiro

- 1 Tipos Enumerados
- 2 Registros
- 3 Redefinição de Tipos

Tipos enumerados

- Para criar uma variável para armazenar um determinado mês de um ano (de janeiro a dezembro), uma das soluções possíveis é utilizar o tipo inteiro e armazenar um número associado àquele mês. Assim, janeiro seria o mês número 1, fevereiro o mês número 2, e assim sucessivamente.
- Entretanto, o código seria mais claro se pudéssemos escrever algo como:

```
mes = jan;
```

Tipos enumerados

- O comando `enum` cria um tipo enumerado: podemos usar nomes/identificadores para um conjunto finito de valores inteiros.
- Sua sintaxe é:

```
enum <tipo> {identificador1, identificador2 ...,  
identificadorN};
```

- Exemplo:

```
/* criamos um novo tipo enumerado chamado meses */  
enum meses {jan, fev, mar, abr, mai, jun,  
            jul, ago, set, out, nov, dez};
```

Usando um tipo enumerado

- O compilador associa o número 0 ao primeiro identificador, 1 ao segundo, etc.
- Variáveis do novo tipo criado são, na realidade, variáveis inteiras.
- Tipos enumerados são usados para deixar o código mais legível.

```
#include <stdio.h>

/* aqui criamos um novo tipo enumerado que pode ser usado por qualquer funcao */
enum meses {jan, fev, mar, abr, mai, jun, jul, ago, set, out, nov, dez};

int main() {
    enum meses a, b; /* aqui criamos 2 variaveis enumeradas do tipo meses */

    a = jan;
    b = jun;
    if (a != b) {
        printf("%d eh um mes diferente de %d\n", a, b);
        /* sera impresso "0 eh um mes diferente de 5" */
    }
    return 0;
}
```

Usando um tipo enumerado

- Note que o primeiro identificador recebeu o valor zero, enquanto demais identificadores receberam valores em sequência.
- Podemos alterar o valor inicial dos identificadores (e conseqüentemente os valores de todos os demais identificadores).

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

```
/* aqui criamos um novo tipo enumerado que pode ser usado por qualquer funcao */  
enum meses {jan = 1, fev, mar, abr, mai, jun, jul, ago, set, out, nov, dez};
```

```
int main() {  
    enum meses a, b; /* aqui criamos 2 variaveis enumeradas do tipo meses */  
  
    a = jan;  
    b = jun;  
    if (a != b) {  
        printf("%d eh um mes diferente de %d\n", a, b);  
        /* sera impresso "1 eh um mes diferente de 6" */  
    }  
    return 0;  
}
```

Outros exemplos de tipos enumerados

- Tipo para armazenar respostas binárias:
`enum resposta {falsa, verdadeira};`
- Tipo para armazenar as estações do ano:
`enum estacao {primavera, verao, outono, inverno};`
- Tipo para armazenar o sexo de uma pessoa:
`enum sexo {masculino, feminino};`
- Tipo para armazenar o naipe de uma carta de baralho:
`enum naipe {ouros, espadas, copas, paus};`
- Tipo para armazenar a direção de navegação:
`enum direcao {N, NE, E, SE, S, SO, O, NO};`

Registros

- Um registro é um mecanismo da linguagem C para agrupar diversas variáveis, que inclusive podem ser de tipos diferentes, mas que, dentro de um contexto, fazem sentido estarem juntas.
- Exemplos de uso de registros:
 - ▶ Registro de alunos para guardar dados como nome, RA, médias de provas e médias de laboratórios.
 - ▶ Registro de pacientes para guardar os dados como nome, endereço e histórico de doenças.

Declarando um registro

- Para criarmos um novo tipo de registro, usamos a palavra chave `struct` da seguinte forma:

```
struct nome_do_tipo_do_registro {
    tipo_1 nome_1;
    tipo_2 nome_2;
    ...
    tipo_n nome_n;
};
```

- Cada `nome_i` é um identificador de um campo do registro que será do tipo `tipo_i`.

Exemplo:

```
struct Aluno {
    char nome[45];
    int idade;
    char sexo;
}; /* estamos criando um novo tipo de registro chamado Aluno */
```

Declarando um registro

- A declaração do registro pode ser feita dentro de uma função (como `main`) ou fora dela. Usualmente, ela é feita fora de qualquer função, como no exemplo abaixo:

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

```
/* declare aqui os registros do seu programa */
```

```
void funcao1() {
```

```
...
```

```
}
```

```
int main () {
```

```
...
```

```
}
```

Declarando um registro

A próxima etapa é declarar uma variável registro do tipo definido, que será usada dentro de seu programa, como no exemplo abaixo:

```
#include <stdio.h>

struct Aluno {
    char nome[45];
    int idade;
    char sexo;
};

int main() {
    /* declarando variaveis registros do tipo Aluno */
    struct Aluno a, b;
    ...
}
```

Utilizando os campos de um registro

- Podemos fazer acesso individualmente aos campos de uma determinada variável registro como se fossem variáveis comuns. A sintaxe é:

`variável_registro.nome_do_campo`

- Os campos individuais de um variável registro têm o mesmo comportamento de qualquer variável do tipo do campo.
 - ▶ Isto significa que todas operações válidas para variáveis de um tipo são válidas para um campo do mesmo tipo.

Utilizando os campos de um registro

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

struct Aluno {
    char nome[45];
    int idade;
    char sexo;
};

int main() {
    struct Aluno a, b;

    strcpy(a.nome, "Helen");
    a.idade = 18;
    a.sexo = 'F';
    strcpy(b.nome, "Dilbert");
    b.idade = 34;
    b.sexo = 'M';

    printf("a.nome = %s, a.idade = %d, a.sexo = %c\n", a.nome, a.idade, a.sexo);
    printf("b.nome = %s, b.idade = %d, b.sexo = %c\n", b.nome, b.idade, b.sexo);
    return 0;
}
```

Lendo e Escrevendo Registros

- A leitura dos campos de um registro deve ser feita campo a campo, como se fossem variáveis independentes.
- O mesmo vale para a escrita, que deve ser feita campo a campo.

...

```
int main() {
    struct Aluno a;

    printf("Digite o nome: ");
    fgets(a.nome, 45, stdin);
    printf("Digite a idade: ");
    scanf("%d", &a.idade);
    printf("Digite o sexo: ");
    setbuf(stdin, 0); /* limpa o buffer de entrada */
    scanf("%c", &a.sexo);

    printf("a.nome = %s, a.idade = %d, a.sexo = %c\n", a.nome, a.idade, a.sexo);
    return 0;
}
```

Atribuição de registros

- Podemos atribuir um registro a outro diretamente:

```
var1_registro = var2_registro;
```

- É feita uma cópia de cada campo do registro.

...

```
int main() {
    struct Aluno a, b;

    printf("Digite o nome: ");
    fgets(a.nome, 45, stdin);
    printf("Digite a idade: ");
    scanf("%d", &a.idade);
    printf("Digite o sexo: ");
    setbuf(stdin, 0); /* limpa o buffer de entrada */
    scanf("%c", &a.sexo);

    b = a;

    printf("b.nome = %s, b.idade = %d, b.sexo = %c\n", b.nome, b.idade, b.sexo);
    return 0;
}
```

Vetor de registros

Pode ser declarado quando necessitamos de diversas cópias de um mesmo tipo de registro (por exemplo, para cadastrar todos os alunos de uma mesma turma).

- Para declarar: `struct Aluno turma[5];`
- Para usar: `turma[indice].campo;`

Vetor de registros

```
#include <stdio.h>

struct Aluno {
    int ra;
    double nota;
};

int main () {
    struct Aluno turma[10];
    int i;
    double nota = 0.0;

    for (i = 0; i < 10; i++) {
        printf("Digite o RA do aluno numero %d: ", i + 1);
        scanf("%d", &turma[i].ra);
        printf("Digite a nota do aluno numero %d: ", i + 1);
        scanf("%lf", &turma[i].nota);
    }
```

Vetor de registros

...

```
/* calcula a media da turma */  
for (i = 0; i < 10; i++)  
    nota = nota + turma[i].nota;  
  
printf("A media da turma eh: %f\n", nota / 10);  
  
return 0;  
}
```

Redefinido um tipo

- 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.
- Por exemplo, em um programa onde manipulamos médias de alunos, todas as variáveis que trabalhassem com nota teriam o tipo `nota` e não `double`.

O comando typedef

- A forma de se fazer isso é utilizando o comando typedef, seguindo a sintaxe abaixo:

```
typedef <tipo_ja_existente> <tipo_novo>;
```

- Usualmente, fazemos essa declaração fora da função main(), embora seja permitido fazer dentro da função também.
- Ex: `typedef float nota;`
Cria um novo tipo, chamado nota, cujas variáveis desse tipo serão pontos flutuantes.

Exemplo de uso do typedef

```
#include <stdio.h>

typedef double nota;

int main() {
    nota p1;

    printf("Digite a nota: ");
    scanf("%lf", &p1);

    printf("A nota digitada foi: %f", p1);

    return 0;
}
```

Exemplo de uso do typedef

- Os usos mais comuns para o comando `typedef` são para redefinições de tipos enumerados e registros.
- Poderíamos redefinir o tipo `enum meses` como simplesmente `meses`:
 - ▶ `typedef enum meses meses;`
- Assim como poderíamos redefinir o tipo `struct Aluno` como simplesmente `Aluno`:
 - ▶ `typedef struct Aluno Aluno;`

Exemplo de uso do typedef

```
#include <stdio.h>

/* definindo o tipo enumerado meses */
enum meses {jan, fev, mar, abr, mai, jun, jul, ago, set, out, nov, dez};
typedef enum meses meses;
/* redefinindo o tipo "enum meses" como "meses" */

int main() {
    meses a, b;

    a = jan;
    b = jun;

    if (a != b) {
        printf("%d eh um mes diferente de %d\n", a, b);
        /* sera impresso "0 eh um mes diferente de 5" */
    }
    return 0;
}
```

Exemplo de uso do typedef

```
#include <stdio.h>

/* definindo o tipo registro Aluno */
struct Aluno {
    int ra;
    double nota;
};
typedef struct Aluno Aluno;
/* redefinindo o tipo "struct Aluno" como "Aluno" */
```

Exemplo de uso do typedef

```
int main () {
    Aluno turma[10];
    int i;
    double nota = 0.0;

    for (i = 0; i < 10; i++) {
        printf("Digite o RA do aluno numero %d: ", i + 1);
        scanf("%d", &turma[i].ra);
        printf("Digite a média do aluno numero %d: ", i + 1);
        scanf("%lf", &turma[i].nota);
    }

    for (i = 0; i < 10; i++)
        nota = nota + turma[i].nota;

    printf("A media da turma eh: %f\n", nota / 10);
    return 0;
}
```

Exemplo de uso de registros

- Suponha que queremos imprimir as informações dos alunos de uma turma (RA e nome) formada por n alunos, onde o valor de n não é conhecido a priori.
- Além da ordem original dos alunos, deseja-se também imprimir as informações ordenadas de outras duas formas:
 - ▶ Em ordem crescente dos números dos RAs.
 - ▶ Em ordem lexicográfica dos nomes.

Exemplo de uso de registros

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

/* definicao do tipo Aluno */
struct Aluno {
    int ra;
    char nome[51];
};
typedef struct Aluno Aluno;

void imprime_turma(Aluno turma[], int n) {
    int i;

    printf("*** Turma ***\n");
    for (i = 0; i < n; i++)
        printf("%06d %s", turma[i].ra, turma[i].nome);
    printf("\n");
}
```

Exemplo de uso de registros

```
void ordena_por_ra(Aluno turma[], int n) {
    int i, j;
    Aluno aux;

    for (i = 1; i < n; i++) {
        aux = turma[i];
        j = i - 1;

        while ((j >= 0) && (turma[j].ra > aux.ra)) {
            turma[j+1] = turma[j];
            j--;
        }

        turma[j+1] = aux;
    }
}
```

Exemplo de uso de registros

```
void ordena_por_nome(Aluno turma[], int n) {
    int i, j;
    Aluno aux;

    for (i = 1; i < n; i++) {
        aux = turma[i];
        j = i - 1;

        while ((j >= 0) && (strcmp(turma[j].nome, aux.nome) > 0)) {
            turma[j+1] = turma[j];
            j--;
        }

        turma[j+1] = aux;
    }
}
```

Exemplo de uso de registros

```
int main () {
    Aluno *turma;
    int n, i;

    printf("Quantos alunos? ");
    scanf("%d", &n);
    turma = malloc(sizeof(Aluno) * n);

    for (i = 0; i < n; i++) {
        printf("Aluno numero %d:\n", i + 1);
        printf("RA: ");
        scanf("%d", &turma[i].ra);
        setbuf(stdin, 0);
        printf("Nome: ");
        fgets(turma[i].nome, 51, stdin);
    }

    ...
}
```

Exemplo de uso de registros

...

```
imprime_turma(turma, n); /* turma na ordem inicial */
```

```
ordena_por_ra(turma, n);
```

```
imprime_turma(turma, n); /* turma ordenada por ra */
```

```
ordena_por_nome(turma, n);
```

```
imprime_turma(turma, n); /* turma ordenada por nome */
```

```
free(turma);
```

```
return 0;
```

```
}
```

Exemplo de uso de registros

*** Turma ***

152034 Ronald Bilius Weasley

161212 Hermione Jean Granger

149826 Harry James Potter

*** Turma ***

149826 Harry James Potter

152034 Ronald Bilius Weasley

161212 Hermione Jean Granger

*** Turma ***

149826 Harry James Potter

161212 Hermione Jean Granger

152034 Ronald Bilius Weasley