MC-102 — Aula 20
Registros

Instituto de Computação – Unicamp

28 de Setembro de 2015

Roteiro

Registros

- Um registro é um mecanismo da linguagem C para agrupar várias 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 os dados: (nome, RA, médias de provas, médias de labs, etc...)
  - Registro de pacientes para guardar os dados: (Nome, endereço, histórico de doenças, etc...)

Declarando um novo tipo de 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_3 nome_3;
  ...
  tipo_n nome_n;
};
```

- Cada *nome_i*, é um identificador que será do tipo *tipo_i* (são declarações de variáveis simples).

Exemplo:

```
struct Aluno{
     char nome[45];
     float nota;
};  //estamos criando um novo tipo "struct Aluno"
```

Declarando um novo tipo de 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, para que qualquer função possa usar dados do tipo de registro criado.

```
#include <stdio.h>

  /* Declare tipos registro aqui */

int main () {
  /* Construa seu programa aqui */
}
```

Declarando um registro

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

```
#include <stdio.h>
struct Aluno{
      char nome[45];
      float nota;
};

int main(){
  struct Aluno a, b; //variáveis a, b são do tipo "struct Aluno"
  ......
 }
```

Utilizando os campos de um registro

- Podemos acessar individualmente os campos de uma determinada variável registro como se fossem variáveis normais. A sintaxe é:

  `variável_registro.nome_do_campo`

- Os campos individuais de um variável registro tem 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];
  float nota;
};

int main(){
  struct Aluno a, b;

  strcpy(a.nome, "Helen");
  a.nota = 8.6;

  strcpy(b.nome, "Dilbert");
  b.nota = 8.2;

  printf("a.nome = %s, a.nota = %f\n", a.nome, a.nota);
  printf("b.nome = %s, b.nota = %f\n", b.nome, b.nota);
}
```

Lendo e Escrevendo Registros

- A leitura dos campos de um registro a partir do teclado deve ser feita campo a campo, como se fossem variáveis independentes.
- A mesma coisa vale para a escrita, que deve ser feita campo a campo.

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

struct Aluno{
  char nome[45];
  float nota;
};

int main(){
  struct Aluno a, b;

  printf("Digite o nome:");
  scanf("%[^\n]", a.nome);
  printf("Digite a nota:");
  scanf("%f", &a.nota);

  printf("a.nome = %s, a.nota = %f\n", a.nome, a.nota);
}
```

Atribuição de registros

- Podemos atribuir um registro a outro diretamente:

```
var1_registro = var2_registro;
```

- Automaticamente é feito uma cópia de cada campo de var2 para var1.

Exemplo:

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

struct Aluno{
  char nome[45];
  float nota;
};
int main(){
  struct Aluno a, b;

  printf("Digite o nome:");
  scanf("%[^\n]", a.nome);
  printf("Digite a nota:");
  scanf("%f", &a.nota);

  b = a;
  printf("b.nome = %s, b.nota = %f\n", b.nome, b.nota);
}
```

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;`

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

struct Aluno{
  char nome[45];
  float nota;
};

int main(){
  struct Aluno turma[5];
  int i;

  for(i=0; i<5; i++){
    printf("Digite o nome:");
    scanf("%[^\n]", turma[i].nome);
    printf("Digite a nota:");
    scanf("%f", &turma[i].nota);
    getchar();//para limpar o buffer
  }
  float media=0;
  for(i=0; i<5; i++){
    media = media + turma[i].nota;
  }

  printf("Media da turma = %f\n", media/5.0);
}
```

- Registros podem ser usados tanto como parâmetros em funções bem como em retorno de funções.
- Neste caso o comportamento de registros é similar ao de tipos básicos.

Exemplo

- Criar aplicação para um cadastro de uma turma com as seguintes funcionalidades:
  - Inclusão de aluno na turma.
  - Exclusão de aluno da turma.
  - Impressão dos alunos na turma.
  - Cálculo da média da turma.
- Usaremos um vetor de Alunos para guardar o cadastro.
- Como alunos podem ser excluídos/incluídos, usaremos a seguinte ideia:
  - Criar campo **usado** no Registro para indicar se aquela posição do vetor está sendo usada ou não com um aluno válido.

Exemplo

- Note no exemplo que devemos iniciar todas os campos **usado** do vetor turma com 0, para indicar que não há dados válidos guardados ali.

```
struct Aluno{
  char nome[80];
  float nota;
  int usado;
};

int main(){
  struct Aluno turma[TAM];
  int i;
  for(i=0;i<TAM;i++)
    turma[i].usado=0; //inicialmente a turma está vazia
  ...
}
```

Exemplo

- Para incluir um novo aluno no cadastro usamos as funções abaixo:

```
struct Aluno leAluno(){
  struct Aluno aux;

  printf("Digite o Nome: ");
  scanf("%s", aux.nome);
  printf("Digite a Nota: ");
  scanf("%f",&aux.nota);
  return aux;
}


void incluiAluno(struct Aluno turma[]){
  printf("Incluindo novo aluno\n");
  int i;
  for(i=0; i<TAM; i++){
    if(turma[i].usado == 0){ //Achou posição vazia
      turma[i] = leAluno();
      turma[i].usado = 1; //Novo aluno na turma
      return;
    }
  }
  printf("Turma Lotada!\n");
}
```

Exemplo

- Para excluir um aluno do cadastro usamos a função abaixo que além de receber como parâmetro o cadastro (vetor **turma**), também recebe o nome do aluno a ser removido.

```
void excluiAluno(struct Aluno turma[], char nomeRem[]){
  printf("Excluindo aluno\n");
  int i;
  for(i=0; i<TAM; i++){
    if(strcmp(turma[i].nome, nomeRem) == 0){
      turma[i].usado = 0; //Posição fica vazia
      return;
    }
  }
  printf("Aluno não encontrado!\n");
}
```

Exemplo

- Para listar alunos da turma, basta percorrer o vetor e imprimir todos os dados das posições onde (**usado == 1**).

```
void listarTurma(struct Aluno turma[]){
  printf("Imprimindo a turma\n");
  int i;
  for(i=0; i<TAM; i++){
    if(turma[i].usado == 1){ //Tem um aluno válido
      printf("Dados de um aluno\n");
      printf("Nome: %s e Nota: %f\n", turma[i].nome, turma[i].nota);
    }
  }
}
```

Exemplo

```
int main(){
  struct Aluno turma[TAM];
  int i;
  for(i=0;i<TAM;i++)
    turma[i].usado=0;
  while(1){
    int op;
    printf("\n\n\nDigite uma opção\n 1-Incluir\n 2-Excluir\n 3-Listar\n 4-Sair\n");
    scanf("%d", &op);

    switch(op){
    case 1:
      incluiAluno(turma);
      break;
    case 2:{
      char nome[80];
      printf("Digite nome do aluno:");
      scanf("%s", nome);
      excluiAluno(turma, nome);
      break;}
    case 3:
      listarTurma(turma);
      break;
    case 4:
      return 0;
    }
  }
}
```

Exercício

- Crie um novo tipo de registro para armazenar coordenadas no plano cartesiano.
- Crie uma função para imprimir um ponto do tipo criado.
- Crie uma função para cada uma destas operações: soma de dois pontos, subtração de dois pontos, multiplicação por um escalar.

Informações Extras: 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 tivessem o tipo `nota`, e não `double`.

Informações Extras: 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.

Informações Extras: 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: %lf",p1);
}
```

Informações Extras: Exemplo de uso do `typedef`

- Mas o uso mais comum para o comando **typedef** é para a redefinição de tipos registro.
- No nosso exemplo de **struct Aluno**, poderíamos redefinir este tipo para algo mais simples como simplesmente **Aluno**:
  - **typedef struct Aluno Aluno;**

```c
#include <stdio.h>

struct Aluno {
  int ra;
  double nota;
};
typedef struct Aluno Aluno; //redefinimos tipo struct Aluno como Aluno

int main (){
  Aluno turma[10];
  int i;   double media;

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

  //calcula a media da turma
  media = 0.0;
  for (i = 0; i < 10; i++) {
    media = media + turma[i].nota;
  }
  media = media/10.0;
  printf("\nA media da turma é: %lf\n",media);
}
```